Exhibit H

Convolutional Codes I: Algebraic Structure

G. DAVID FORNEY, JR., MEMBER, IEEE

Abstract—**A convolutional encoder is defined as any constant linear sequential circuit. The associated code is the set of all output sequences resulting from any set of input sequences beginning at any time. Encoders are called equivalent if they generate the same code. The invariant factor theorem is used to determine when a convolutional encoder has a feedback-free inverse, and the minimum delay of any inverse. All encoders are shown to be equivalent to minimal encoders, which are feedback-free encoders with feedback-free delay-free inverses, and which can be realized in the conventional manner with as few memory elements as any equivalent encoder. Minimal encoders are shown to be immune to catastrophic error propagation and, in fact, to lead in a certain sense to the shortest decoded error sequences possible per error event. In two appendices, we introduce dual codes and syndromes, and show that a minimal encoder for a dual code has exactly the complexity of the original encoder; we show that systematic encoders with feedback form a canonical class, and compare this class to the minimal class.**

I. Introduction

BLOCK CODES were the earliest type of codes to be investigated, and remain the subject of the overwhelming bulk of the coding literature. On the other hand, convolutional codes have proved to be equal or superior to block codes in performance in nearly every type of practical application, and are generally simpler than comparable block codes in implementation. This anomaly is due largely to the difficulty of analyzing convolutional codes, as compared to block codes. It is the intent of this series to stimulate increased theoretical interest in convolutional codes by review and clarification of known results and introduction of new ones. We hope, first, to advance the understanding of convolutional codes and tools useful in their analysis; second, to motivate further work by showing that in every case in which block codes and convolutional codes can be directly compared theoretically, the convolutional are as good or better.

Two converging lines of development have generated interest in an algebraic approach to convolutional codes. On the one hand, the success of algebraic methods in generating classes of good block codes suggests that constructive methods of generating good convolutional codes might be developed through use of algebraic structures. Correspondingly one might expect that powerful decoding techniques based on such structures might be discovered. (Fortunately, good codes and decoding methods not relying on such constructions are already known.) On the other hand, the usefulness of regarding convolutional encoders as linear sequential circuits has begun to become evident, as in the observation of Omura [1] and others that the Viterbi [2] maximum-likelihood decoding algorithm is the dynamic programming solution to a certain control problem, and in the observation of Massey and his colleagues [3]–[8] that certain questions concerning error propagation are related to questions concerning the invertibility of linear systems. As the theory of finite-dimensional linear systems is seen increasingly as essentially algebraic, we have another motive for examining convolutional encoders in an algebraic context.

Our result is a series of structure theorems that dissect the structure of convolutional codes rather completely, mainly through use of the invariant factor theorem. We arrive at a class of canonical encoders capable of generating any convolutional code, and endowed with all the desirable properties one might wish, except that in general they are not systematic. (The alternate canonical class of systematic encoders with feedback is discussed in Appendix II.) The results do not seem to suggest any constructive methods of generating good codes, and say little new in particular about the important class of rate-$1/n$ codes, except for putting known results in a more general context. It appears that the results obtained here for convolutional codes correspond to block-code results ([9], ch. 3).

Manuscript received December 18, 1969. Part of this paper was presented at the Internatl. Information Theory Symposium, Ellenville, N. Y., January 27–31, 1969.
The author is with Codex Corporation, Newton, Mass. 02158.

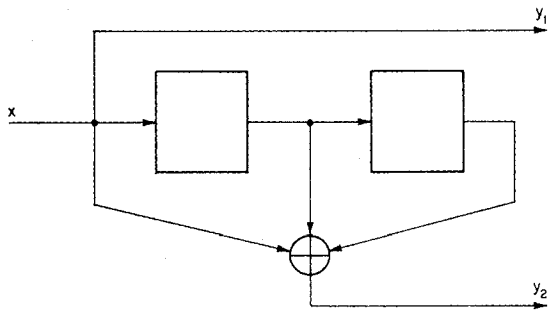
II. Problem Formulation

We are purposefully going to take a rather long time getting to our main results. Most of this time will be spent in definitions and statements of fundamental results in convolutional coding theory, linear system theory, and algebra. It is a truism that when dealing with fundamentals, once the problem is stated correctly, the results are easy. We feel it is important that the right formulation of the problem (like justice) not only be done, but be seen to be done, in the eyes of readers who may have backgrounds in any of the three areas noted.

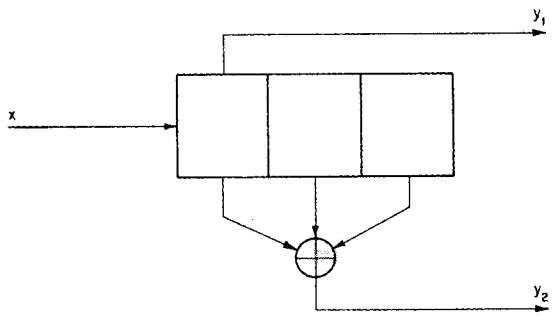
After exhibiting a simple convolutional encoder for motivation, we move to a general definition of convolutional encoders, which we see amount to general finite-state time-invariant linear circuits. We discuss the decoding problem, which leads to a definition of convolutional encoder equivalence and to certain desirable code properties. Along the way certain algebraic artifacts will intrude into the discussion; in the final introductory section we collect the algebra we need, which is centered on the invariant factor theorem.

Convolutional Encoder

Fig. 1(a) shows a simple binary systematic rate-1/2 convolutional encoder of constraint length 2. The input to this encoder is a binary sequence



(a)



(b)

Fig. 1. (a) A rate-1/2 systematic convolutional encoder. (b) Alternate representation.

$$x = (\cdots, x_{-1}, x_0, x_1, \cdots).$$

The outputs are two binary sequences y_1 and y_2 (hence the rate is 1/2). The first output sequence y_1 is simply equal to the input x (hence the code is systematic). The elements y_{2i} of the second sequence y_2 are given by

$$y_{2i} = x_i \oplus x_{i-1} \oplus x_{i-2},$$

where $\oplus$ denotes modulo 2 addition. Therefore, the encoder must save 2 past information bits, so we say the constraint length is 2.

Sometimes it is convenient to draw the encoder as in Fig. 1(b), with the output a function of the memory contents only. (This corresponds to the distinction in automata theory between Moore and Mealy machines.) Some authors would therefore say this code had constraint length 3, since the outputs are a function of a span of 3 input bits. Others measure constraint length in terms of output bits and would assign this code a constraint length of 6. Our definition of constraint length is chosen to coincide with the number of memory elements in a minimal realization.

The term "convolutional" comes from the observation that the output sequences can be regarded as a convolution of the input sequence with certain generator sequences. With the input and output sequences, we associate sequences in the delay operator D (D transforms):

$$x(D) = \cdots + x_{-1}D^{-1} + x_0 + x_1D + x_2D^2 + \cdots$$
$$y_1(D) = \cdots + y_{1,-1}D^{-1} + y_{10} + y_{11}D + y_{12}D^2 + \cdots$$
$$y_2(D) = \cdots + y_{2,-1}D^{-1} + y_{20} + y_{21}D + y_{22}D^2 + \cdots.$$

(The delay operator D corresponds to z^{-1} in sampled-data theory, but is purely an indeterminate or place-holder, whereas z is a complex variable.) Now the input/output relationships are expressed concisely as

$$y_1(D) = g_1(D)x(D)$$
$$y_2(D) = g_2(D)x(D),$$

where the generator polynomials $g_1(D)$ and $g_2(D)$ are

$$g_1(D) = 1$$
$$g_2(D) = 1 + D + D^2,$$

and ordinary sequence multiplication with coefficient operations modulo 2 and collection of like powers of D is implied.

Similarly, we can define a general (n, k) conventional convolutional encoder by a matrix of generator polynomials $g_{ij}(D)$, $1 \leq i \leq k$, $1 \leq j \leq n$, with coefficients in some finite field F. There are k-input sequences $x_i(D)$, n-output sequences $y_j(D)$, each a sequence of symbols from F, with input/output relations given by

$$y_j(D) = \sum_{i=1}^{k} x_i(D)g_{ij}(D),$$

again with all operations in F. If we define the constraint length for input i as

$$\nu_i = \max_{1 \leq j \leq n} [\deg g_{ij}(D)],$$

then the general conventional encoder can be realized by k shift registers, the ith of length ν_i, with the outputs formed as linear combinations in F on the appropriate shift register contents. We call this the obvious realization, and note that the number of memory elements required is equal to the overall constraint length, defined as the sum of constraint lengths

$$\nu = \sum_{i=1}^{k} \nu_i.$$

For notational convenience we shall generally suppress the parenthetical D in our subsequent references to sequences; thus x_i means $x_i(D)$, $y_j = y_j(D)$, and so forth, where the fact that a letter represents a sequence (transform) should be clear from the context.

Convolutional Encoder—General Definition

The encoders of the previous section are linear sequential circuits. We now consider all finite-state time-invariant linear sequential circuits as candidates for convolutional encoders.

Definition 1: An (n, k) convolutional encoder over a finite field F is a k-input n-output constant linear causal finite-state sequential circuit.

Let us dissect this definition.

K-input: There are k discrete-time input sequences x_i, each with elements from F. We write the inputs as the row vector x. Sequences must start at some finite time and may or may not end. (Then we can represent a sequence such as $1 + D + D^2 + \cdots$ by a ratio of poly-

nomials such as $1/(1 + D)$, without encountering the ambiguity $1/(1 + D) = D^{-1} + D^{-2} + \cdots$.) If a sequence x_i "starts" at time d (if the first nonzero element is x_{id}), we say it has delay d, del $x_i = d$, and if it ends at time d', we say it has degree d', deg $x_i = d'$, in analogy to the degree of a polynomial. Similarly, we define the delay and degree of a vector of sequences as the minimum delay or maximum degree of the component sequences: del x = min del x_i, deg x = max deg x_i. A finite sequence has both a beginning and an end. (Note that most authors consider only sequences starting at time 0 or later. It turns out that this assumption clutters up the analysis without compensating benefits.)

N-output: There are n-output sequences y_j, each with elements from F, which we write as the row vector $\boldsymbol{y}$. The encoder is characterized by the map G, which maps any vector of input sequences $\boldsymbol{x}$ into some output vector $\boldsymbol{y}$, which we can write in the functional form

$$\boldsymbol{y} = G(\boldsymbol{x}).$$

Constant (Time-Invariant): If all input sequences are shifted in time, all output sequences are correspondingly shifted. In delay-operator notation,

$$G(D^n\boldsymbol{x}) = D^nG(\boldsymbol{x}) \qquad n \text{ any integer.}$$

(Note that most probabilistic analyses of convolutional codes have been forced to assume nonconstancy to obtain ensembles of encoders with enough randomness to prove theorems.)

Linear: The output resulting from the superposition of two inputs is the superposition of the two outputs that would result from the inputs separately, and the output "scales" with the input. That is,

$$G(\boldsymbol{x}_1 + \boldsymbol{x}_2) = G(\boldsymbol{x}_1) + G(\boldsymbol{x}_2)$$
$$G(\alpha\boldsymbol{x}_1) = \alpha G(\boldsymbol{x}_1) \qquad \alpha \in F.$$

It is easy to see that constancy and linearity together give the broader linearity condition

$$G(\alpha\boldsymbol{x}_1) = \alpha G(\boldsymbol{x}_1),$$

where α is any sequence of elements of F in the delay operator D. Furthermore, they imply a transfer-function representation for the encoder. For let $\boldsymbol{\varepsilon}_i$, $1 \leq i \leq k$, be the unit inputs in which the ith input at time 0 is 1, and all other inputs are 0. Let the generators $\boldsymbol{g}_i$ be defined as the corresponding outputs (impulse responses):

$$\boldsymbol{g}_i = G(\boldsymbol{\varepsilon}_i) \qquad 1 \leq i \leq k.$$

Then since any input $\boldsymbol{x}$ can be written

$$\boldsymbol{x} = \sum_{i=1}^{k} x_i\boldsymbol{\varepsilon}_i,$$

we have by linearity

$$\boldsymbol{y} = G(\boldsymbol{x}) = \sum_{i=1}^{k} x_i\boldsymbol{g}_i.$$

Thus we can define a $k \times n$ transfer function matrix G, whose rows are the generators $\boldsymbol{g}_i$, such that the input/output relationship is

$$\boldsymbol{y} = \boldsymbol{x}G.$$

Therefore from this point we use the matrix notation $\boldsymbol{y} = \boldsymbol{x}G$ in preference to the functional notation $\boldsymbol{y} = G(\boldsymbol{x})$.

Finally, this definition implies that a zero input gives a zero output so that the encoder may not have any transient response (nonzero starting state).

Causal: If the nonzero inputs start at time t, then the nonzero outputs start at time $t' \geq t$. Since the unit inputs start at time 0, this implies that all generators start at time 0 or later. As sequences, all generators must therefore have all negative coefficients equal to 0, or del $\boldsymbol{g}_i \geq 0$. Conversely, the condition that all generators satisfy del $\boldsymbol{g}_i \geq 0$ implies

$$\begin{aligned}\text{del } \boldsymbol{y} &= \text{del} \sum_{i=1}^{k} x_i\boldsymbol{g}_i \\ &\geq \min_{1\leq i\leq k} [\text{del } x_i + \text{del } \boldsymbol{g}_i] \\ &\geq \text{del } \boldsymbol{x},\end{aligned}$$

causality.

Finite-state: The encoder shall have only a finite number of memory elements, each capable of assuming a finite number of values. The physical state of an encoder at any time is the contents of its memory elements; thus there are only a finite number of physical states. A more abstract definition of the state of an encoder at any time is the following: the state $\boldsymbol{s}$ of an encoder at time t^- is the sequence of outputs at time t and later if there are no nonzero inputs at time t or later. Clearly the number of states so defined for any fixed t is less than or equal to the number of physical states, since causality implies that each physical state gives some definite sequence, perhaps not unique. Thus an encoder with a finite number of physical states must have a finite number of abstract states as well.

By studying the abstract states, we develop further restrictions on the generators $\boldsymbol{g}_i$. Let us examine the set of possible states at time 1^-, which we call the state space Σ. (By constancy the state spaces at all times are isomorphic.) Let P be the projection operator that truncates sequences to end at time 0, and Q the complementary projection operator $1 - P$ that truncates sequences to start at time 1:

$$\boldsymbol{x}P = x_dD^d + \cdots + x_{-1}D^{-1} + x_0$$
$$\boldsymbol{x}Q = x_1D^2 + x_2D^2 + \cdots.$$

Then any input $\boldsymbol{x}$ is associated with a state at time 1^- given by

$$\boldsymbol{s} = \boldsymbol{x}PGQ;$$

conversely, any state in Σ can be so expressed using any input giving that state. Now the state space Σ is seen to satisfy the conditions to be a vector space over the field F, for P, G, and Q are all linear over F; that is, if

$$s_1 = x_1PGQ,$$

and

$$s_2 = x_2PGQ,$$

then the combinations

$$s_1 + s_2 = (x_1 + x_2)PGQ$$
$$\alpha s_1 = (\alpha x_1)PGQ \qquad \alpha \in F,$$

are also states. As a vector space, Σ therefore has a finite dimension dim Σ, or else the number of states $q^{\dim \Sigma}$, where q is the number of elements in the finite field, would be infinite.

Consider now for simplicity a single-input single-output linear sequential circuit with transfer function g, so $y = xg$. Let s_d be the state obtained from a unit input at time $-d$:

$$s_d = D^{-d}gQ \qquad d \geq 0.$$

If the state space has finite dimension dim Σ, then at most dim Σ of these states can be linearly independent, and in particular there is some linear dependence relation (with coefficients ψ_d in F) between the first dim $\Sigma + 1$ of these states:

$$0 = \sum_{d=0}^{\dim \Sigma} \psi_d s_d$$
$$= \psi(D^{-1})gQ,$$

where

$$\psi(D^{-1}) = \sum_{d=0}^{\dim \Sigma} \psi_d D^{-d}$$

is some nonzero polynomial over F of degree dim Σ or less in the indeterminate D^{-1}. In order for $\psi(D^{-1})g$ to be 0 at time 1 and later, g itself must be equal to a ratio of polynomials $h(D^{-1})/\psi(D^{-1})$, with the degree of the numerator $h(D^{-1})$ not greater than that of the denominator $\psi(D^{-1})$ in order that g be causal, del $g \geq 0$. Any sequence g that can be so expressed will be called a realizable function, realizable meaning both causal and finite. Clearly a realizable function can also be expressed (by multiplying numerator and denominator by $D_{\deg \psi}$) as a ratio of polynomials in D rather than D^{-1}, in which case the condition deg $h \leq$ deg ψ is transformed to the condition that D not be a factor of the denominator after reduction to lowest terms.

We shall always assume that the expression $h(D^{-1})/\psi(D^{-1})$ of a realizable function g has been reduced to lowest terms and has a monic denominator ($\psi_{\deg \psi} = 1$). A canonical realization of such a realizable function is shown in Fig. 2. The realization involves a feedback shift register with deg ψ memory elements storing elements of F, which essentially performs long division; if h/ψ is in lowest terms then the dimension of the state-space dim Σ is also equal to deg ψ.

A convolutional encoder is then any linear sequential circuit whose transfer-function matrix G is realizable, that is, has components that are realizable functions. It is clear that a brute-force finite-state realization of such an encoder always exists, since one can simply realize the kn components and sum their outputs; in Appendix II we discuss the result of realization theory that shows how to construct a canonical realization, namely, one with a number of memory elements equal to the dimension of the abstract state space. We see that the only respect in which this definition is more general than that of a conventional convolutional encoder is that realizable generators in general involve feedback and, as sequences, have infinite length. It might seem that getting an infinite-length generator sequence from a finite-state encoder is a good bargain, but we shall see in the sequel that in fact feedback buys nothing.

Communications Context

So far we have defined a general convolutional encoder merely as a general realizable linear sequential circuit, and developed basic concepts in linear system theory. It is only when we put the encoder in a communications context that questions unlike those posed in linear system theory arise. The most important new concept involves a definition of encoder equivalence very different from the linear-system-theoretic one.

Consider then the use of a convolutional encoder in a communications system, shown in Fig. 3. From the k-input sequences x, called information sequences, the encoder G generates a set of n-output sequences y, called a codeword, which is transmitted over some noisy channel. The received data, whatever their form, are denoted by r; a decoder operates on r in some way to produce k decoded sequences $\hat{x}$, preferably not too different from x.

We see immediately that all is lost if the encoder map G is not one-to-one, for then even if there were no noise in the channel the decoder would make mistakes. In fact, by linearity, if $y = x_1G = x_2G$ for two different inputs, then for any output y' there are at least two inputs x' and $x' + x_1 - x_2$ such that

$$y' = x'G = (x' + x_1 - x_2)G;$$

and by constancy the difference between the inputs may be made to extend over all time by concatenating them if they are not infinite already, so that the probability of decoding error would be at least $\frac{1}{2}$. We therefore require that the encoder map be one-to-one in any useful convolutional encoder. There is therefore some *inverse* map G^{-1}, obviously linear and constant, which takes outputs y back to the unique corresponding input x,

$$xGG^{-1} = x \qquad \text{for all } x;$$
$$GG^{-1} = I_k,$$

where I_k is the identity transformation. (Here we have used an $n \times k$ transfer-function matrix representation for G^{-1}, as we may, since G^{-1} is constant and linear; in matrix terminology G^{-1} is a right inverse.) Of course this inverse map may not be realizable, but we shall show below that when there is any inverse there is always a

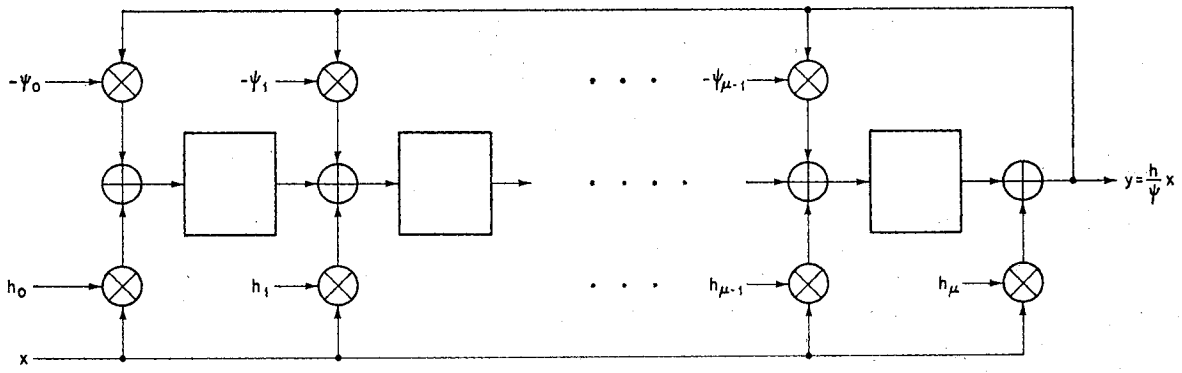


Fig. 2. Canonical realization of the transfer function $h(D^{-1})/\psi(D^{-1})$.

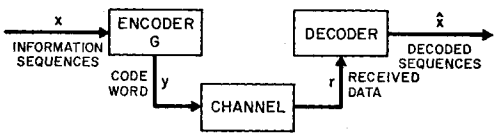


Fig. 3. Communications system using convolutional coding.

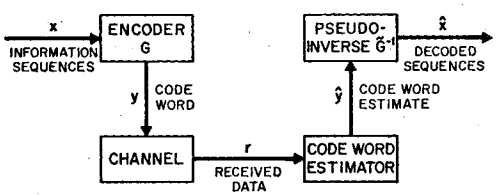


Fig. 4. Same system with decoder in two parts.

realizable pseudo-inverse $\tilde{G}^{-1}$ such that $G\tilde{G}^{-1} = I_k D^d$ for some $d \geq 0$. $\tilde{G}^{-1}$ is also called an inverse with delay d, and G is sometimes called information-lossless of order d.

Returning to Fig. 3, we now split the decoder conceptually into two parts, a codeword estimator that produces from the received data r some codeword estimate $\hat{\mathbf{y}}$, followed by a realizable pseudo-inverse $\tilde{G}^{-1}$ that assigns to the codeword estimate the appropriate information sequences $\hat{\mathbf{x}}$ (see Fig. 4). In practice a decoder is usually not realized in these two pieces, but it is clear that since all the information about the information sequences $\boldsymbol{x}$ comes through $\boldsymbol{y}$, the decoder can do no better estimating $\boldsymbol{x}$ directly than by estimating $\boldsymbol{y}$ and making the one-to-one correspondence to $\boldsymbol{x}$. (In fact, any decoder can be made into a codeword estimator by appending an encoder G.)

When G is one-to-one, as long as the codeword estimator makes no errors, there will be no error in the decoded sequences. However, even in the best-regulated communications systems, decoding errors will sometimes occur. We define the error sequences as the difference between the estimated codeword $\hat{\mathbf{y}}$ and the codeword $\boldsymbol{y}$ actually sent:

$$\boldsymbol{e} = \hat{\mathbf{y}} - \boldsymbol{y}.$$

Correspondingly the information errors $\boldsymbol{e}_x$ are defined as the difference between the decoded sequences $\hat{\mathbf{x}}$ and the information sequences $\boldsymbol{x}$, with allowance for the pseudo-inverse delay d:

$$\begin{aligned}\boldsymbol{e}_x &= \hat{\mathbf{x}}D^{-d} - \boldsymbol{x} \\ &= \hat{\mathbf{y}}\tilde{G}^{-1}D^{-d} - \boldsymbol{y}G^{-1} \\ &= \boldsymbol{e}G^{-1}.\end{aligned}$$

The one-to-one correspondence between these two definitions is exhibited explicitly through the inverse G^{-1}.

As is implicit in our terminology, we consider the error sequences $\boldsymbol{e}$ as the more basic of the two.

Since the error sequences are the difference between two codewords, $\boldsymbol{e}$ is itself a codeword, in fact the codeword generated by $\boldsymbol{e}_x$:

$$\boldsymbol{e} = \boldsymbol{e}_x G.$$

We make a decomposition of $\boldsymbol{e}$ into short codewords, called error events, as follows. Start at some time when the codeword estimator has been decoding correctly. An error event starts at the first subsequent time at which $\boldsymbol{e}$ is nonzero. The error event stops at the first subsequent time when the error sequences in the event form a finite codeword, after which the decoder will be decoding correctly again. Thus we express $\boldsymbol{e}$ as a sum of nonoverlapping finite codewords, the error events.

Implicit in the above analysis is the assumption that infinite error events do not occur. Such a possibility is not excluded in principle, but can generally be disregarded, on the basis of the following plausibility argument. If two codewords differ in an infinite number of places, then as time goes on, we can expect the evidence in the received data r to build up in favor of the correct word, with probability approaching 1. A well-designed codeword estimator will use this information efficiently enough that very long error events have very small probabilities and infinite error events have probability 0. More precisely, the average error-event length should be small, at least for channel noise, which is in some sense small, so that most of the time the decoder is decoding correctly. We say that any codeword estimator or decoder not satisfying this assumption is subject to ordinary error propagation, and exclude it from the class of useful decoders. Note that any decoder that is capable of starting to decode correctly sooner or later, regardless of what time it is

turned on, can be made immune to ordinary error propagation simply by restarting whenever it detects it is in trouble, or even periodically.

We owe to Massey and Sain [4] the observation that if there is any infinite information sequence x_0 such that the corresponding codeword $y_0 = x_0G$ is finite, then even in the absence of ordinary error propagation decoding catastrophes can occur. For there will generally be a nonzero probability for the finite error event $e = y_0$ to occur, which will lead to infinite information errors $e_x = x_0$. Thus $\hat{x}$ will differ from the original information sequences x in an infinite number of places, even if no further decoding errors are made by the codeword estimator. (In fact, the only chance to stop this propagation is to make another error.) Massey and Sain call this catastrophic error propagation. Since $\tilde{G}^{-1}$ must supply the infinite output x_0 in response to the finite input y_0, it must have internal feedback for the above situation to occur. We therefore require that any useful encoder must not only have a realizable pseudo-inverse $\tilde{G}^{-1}$, but one that is feedback-free. (We see later that if G has no such inverse, then there is indeed an infinite input leading to a finite output.)

We come at last to our most important observations (see also [3]). The codeword estimator dominates both complexity and performance of the system: complexity, because both G and $\tilde{G}^{-1}$ represent simple one-to-one (and in fact linear) maps, while the codeword estimator map from received data r to codeword $\hat{y}$ is many-to-one; performance, because in the absence of catastrophic error propagation the probability of decoding error is equal to the probability of an error event multiplied by the average decoding errors per error event, with the former generally dominant and the latter only changed by small factors for reasonable pseudo-inverses $\tilde{G}^{-1}$ (in fact, in many practical applications the probability of error event is the significant quantity, rather than the probability of decoding error). But the performance and complexity of the codeword estimator depend only on the set of codewords y, not on the input/output relationship specified by G (assuming that all x and hence y are equally likely, or at least that the codeword estimator does not use codeword probabilities, as is true, for example, in maximum-likelihood estimation). Therefore it is natural to assert that two encoders generating the same set of codewords are essentially equivalent in a communications context. So we give the following definitions.

Definition 2: The *code* generated by a convolutional encoder G is the set of all codewords $y = xG$, where the k inputs x are any sequences.

Definition 3: Two encoders are *equivalent* if they generate the same code.

These definitions free us to seek out the encoder in any equivalence class of encoders that has the most desirable properties. We have already seen the desirability of having a feedback-free realizable pseudo-inverse $\tilde{G}^{-1}$. Our main result is that any code can be generated by an encoder G with such a $\tilde{G}^{-1}$, and in fact with the following properties.

1) G has a realizable feedback-free zero-delay inverse G^{-1}.
2) G is itself conventional (feedback-free).
3) The obvious realization of G requires as few memory elements as any equivalent encoder.
4) Short codewords are associated with short information sequences, in a sense to be made precise later.

In the context of linear system theory, the study of convolutional encoders under equivalence can be viewed as the study of those properties of linear systems that belong to the output space alone, or of the invariants over the class of all systems with the same output spaces.

Algebra

We assume that the reader has an algebraic background roughly at the level of the introductory chapters of Peterson [9]. He will therefore be familiar with the notion of a field as a collection of objects that can be added, subtracted, multiplied, and divided with the usual associative, distributive, and commutative rules; he will also know what is meant by a vector space over a field. Further, he will understand that a (commutative) ring is a collection of objects with all the properties of fields except division. He should also recall that the set of all polynomials in D with coefficients in a field F, written conventionally as $F[D]$, is a ring.

The polynomial ring $F[D]$ is actually an example of the very best kind of ring, a principal ideal domain. The set of integers is another such example. Without giving the technical definition of such a ring, we can describe some of its more convenient properties. In a principal ideal domain, certain elements r, including 1, have inverses r^{-1} such that $rr^{-1} = 1$; such elements are called units. The unit integers are ± 1, and the unit polynomials are those of degree 0, namely, the nonzero elements of F. Those elements that are not units can be uniquely factored into products of primes, up to units; a prime is an element that has no factor but itself, up to units. (The ambiguity induced by the units is eliminated by some convention: the prime integers are taken to be the positive primes, while the prime polynominals are taken to be *monic* irreducible polynomials, where monic means having highest order coefficient equal to 1.) It follows that we can cancel: if $ab = ac$, then $b = c$; this is almost as good as division. Further, we have the notion of the greatest common divisor of a set of elements as the greatest product of primes that divides all elements of the set, again made unique by the conventional designation of primes.

Other principal ideal domains have already occurred in our discussions. In general, if R is any principal ideal domain and S is any multiplicative subset—namely, a group of elements containing 1 but not 0 such that if $a \in S$, $b \in S$, then $ab \in S$—then the ring of fractions $S^{-1}R$ consisting of the elements r/s where $r \in R$ and $s \in S$ is a principal ideal domain. (All elements of R that are in S are thereby given inverses and become units.) Letting R be the polynomials $F[D]$, we have the following examples.

1) Let S consist of all nonzero polynomials. Then $S^{-1}R$ consists of all ratios of polynomials with the denominator nonzero, which are called the rational functions, written conventionally $F(D)$. Obviously, in $F(D)$ all nonzero elements are invertible, so $F(D)$ is actually a field, called the field of quotients of $F[D]$.

2) Let S consist of all nonnegative powers of D, including $D^0 = 1$. Then $S^{-1}R$ consists of elements $D^{-n}f(D)$, where $f(D)$ is a polynomial; in other words, $S^{-1}R$ is the set of finite sequences $F_{fi}(D)$. Clearly all the irreducible polynomials except D remain as primes in $F_{fi}(D)$.

3) Let S consist of all polynomials with nonzero constant term; that is, not divisible by D. Then $S^{-1}R$ consists of ratios of polynomials in D with a nonzero constant term in the denominator. We saw earlier that these are precisely the generators realizable by causal finite-state systems; these are therefore called the realizable functions $F_{rs}(D)$. Note that in $F_{rs}(D)$, D is the only prime element.

In addition to the above, we shall be considering the ring of polynomials in D^{-1}, $F[D^{-1}]$. We originally obtained the realizable functions as ratios of polynomials in D^{-1} with the degree of the numerator less than or equal to the degree of the denominator; the realizable functions thus form a ring of fractions of $F[D^{-1}]$, but not of the type $S^{-1}R$. In this ring the only prime is expressed as $(1/D^{-1})$.

We also use the ring containing all sequences x such that $\text{del } x \geq 0$, which in algebra is called the formal power series in D and denoted conventionally as $F[[D]]$; $F[[D]]$ is also a principal ideal domain, whose only prime is D.

If G is a matrix of field elements, then it generates a vector space over the field. If it is a matrix of ring elements, then it generates a module over the ring. (A module is defined precisely like a vector space, except the scalars are in a ring rather than a field. This is the difference between block and convolutional codes.) The main theorem concerning modules over a principal ideal domain—some would say the only theorem—is a structure theorem, which, when applied to matrices G, is called the invariant-factor theorem. This theorem alone, when extended and applied to different rings, yields most of our results.

Invariant-Factor Theorem: Let R be a principal ideal domain and let G be a $k \times n$ R-matrix. Then G has an invariant-factor decomposition

$$G = A\Gamma B,$$

where A is a square $k \times k$ R-matrix with unit determinant, hence with an R-matrix inverse A^{-1}; B is a square $n \times n$ R-matrix with R-matrix inverse B^{-1}; and Γ is a $k \times n$ diagonal matrix, whose diagonal elements γ_i, $1 \leq i \leq k$, are called the invariant factors of G with respect to R. The invariant factors are unique, and are computable as follows: let Δ_i be the greatest common divisor of the $i \times i$ subdeterminants (minors) of G, with $\Delta_0 = 1$ by convention; then $\gamma_i = \Delta_i/\Delta_{i-1}$. We have that γ_i divides γ_{i+1} if γ_{i+1} is not zero, $1 \leq i \leq k-1$. The matrices A and B can be obtained by a computational algorithm (sketched below); they are not in general unique. Finally, if there is any decomposition $G = A\Gamma B$ such that A and B are invertible R-matrices and Γ is a diagonal matrix with $\gamma_i \mid \gamma_{i+1}$ or $\gamma_{i+1} = 0$, then the γ_i are the invariant factors of G with respect to R.

Sketch of Proof [10]: G is said (in this context only) to be equivalent to G' if $G = AG'B$, where A and B are square $k \times k$ and $n \times n$ R-matrices with unit determinants; the assertion of the theorem is that G is equivalent to a diagonal matrix Γ that is unique under the specified conditions on the γ_i. Since any such A can be represented as the product of elementary row operations, and B of elementary column operations (interchange of rows (columns), multiplication of any row (column) by a unit in R, addition of any R-multiple of any row (column) to another), it can be shown that the Δ_i are preserved under equivalence. In particular, therefore, Δ_1 divides all elements of all equivalent matrices. We will now show that there exists an equivalent matrix in which some element divides all other elements, hence is equal to Δ_1 up to units. Let G not be already of this form, and let α and β be two nonzero elements in the same row or column such that α does not divide β. (If there is no element in the same row or column as α not divisible by α, there is some such β in some other column, and this column can be added to the column containing α to give an equivalent matrix for which the prescription above can be satisfied.) By row or column permutations α may be placed in the upper-left corner and β in the second entry of the first row or column; we assume column for definiteness. Now there exist x and y such that $\alpha x + \rho y = \delta$, where δ is the greatest common divisor of α and β, and has fewer prime factors than α since $\alpha \nmid \beta$, $\delta \neq \alpha$. The row transformation below then preserves equivalence while replacing α by δ:

$$\left[\begin{array}{cc|c} x & y & 0 \\ -\beta/\delta & \alpha/\delta & \\ \hline 0 & & I_{k-2} \end{array}\right] \cdot \left[\begin{array}{c|c} \alpha \cdot & \cdot \\ \beta \cdot & \\ \hline \cdot & \cdot \end{array}\right] = \left[\begin{array}{c|c} \delta \cdot & \cdot \\ 0 \cdot & \\ \hline \cdot & \cdot \end{array}\right].$$

If δ does not now divide all elements of the equivalent matrix, the construction can be repeated and δ replaced by some δ' with fewer prime factors. This descending chain can therefore terminate only with a δ that does divide all elements of the equivalent matrix. Since $\delta = \Delta_1 = \gamma_1$, up to units, multiplication of the top row by a unit puts γ_1 in the upper-left corner, and the whole first row and column can be cleared to zero by transformations of the above type (with $x = 1$, $y = 0$), giving the equivalent matrix

$$G' = \left[\begin{array}{c|c} \gamma_1 & 0 \\ \hline 0 & G_1 \end{array}\right]$$

where γ_1 divides every element of G_1. Similarly, G_1 is equivalent to a matrix G_1' of the same form, so

$$G \cong \left[\begin{array}{cc|c} \gamma_1 & 0 & 0 \\ 0 & \gamma_2 & \\ \hline & 0 & G_2 \end{array}\right]$$

where γ_1 divides γ_2 and γ_2 divides all elements of G_2. Continuing in this way, we arrive at a diagonal matrix Γ meeting the conditions of the theorem. Its uniqueness and the formula for the γ_i are obtained from the relationship $\Delta_i = \prod_{i' \leq i} \gamma_{i'}$. Q.E.D.

The invariant-factor decomposition involves a similarity transformation in some respects reminiscent of diagonalizing transformations of square matrices over a field; the invariant factors have some of the character of eigenvalues. The analogy cannot be pressed very far however.

The extension of the invariant-factor theorem to rings of fractions is immediate.

Invariant-Factor Theorem (Extension): Let R be a principal ideal domain and let Q be a ring of fractions of R. Let G be a $k \times n$ Q-matrix. Let ψ be the least common multiple of all denominators in G; then ψG is an R-matrix. Consequently ψG has an invariant-factor decomposition

$$\psi G = A \Gamma' B.$$

Dividing through by ψ, we obtain an invariant-factor decomposition of the Q-matrix G with respect to R

$$G = A \Gamma B$$

where

$$\Gamma = \Gamma' / \psi.$$

Here A and B are R-matrices with R-matrix inverses A^{-1} and B^{-1}. The diagonal elements γ_i of Γ are elements of Q uniquely determined as $\gamma_i = \gamma'_i/\psi = \alpha_i/\beta_i$, where α_i and β_i are obtained by canceling common factors in γ'_i and ψ, gcd $(\alpha_i, \beta_i) = 1$. Since $\gamma'_i \mid \gamma'_{i+1}$ if $\gamma'_{i+1} \neq 0$, we have that $\alpha_i \mid \alpha_{i+1}$ if $\alpha_{i+1} \neq 0$ and $\beta_{i+1} \mid \beta_i$, $1 \leq i \leq k-1$. Explicitly, if ψ_i is the least common multiple of the denominators of the $i \times i$ subdeterminants of G, if θ_i is the greatest common divisor of the numerators, and $\Delta_i = \theta_i/\psi_i$ with $\Delta_0 = 1$ by convention, then

$$\gamma_i = \alpha_i/\beta_i = \Delta_i/\Delta_{i-1} \qquad 1 \leq i \leq k.$$

The γ_i are called the invariant factors of the Q-matrix G with respect to R. Finally, if there exists any $G = A \Gamma B$ satisfying the above conditions, then the diagonal terms of Γ are the invariant factors of G with respect to R.

We may picture an invariant factor decomposition more concretely as follows. Let a $k \times k$ scrambler A be defined as a $k \times k$ R-matrix with an R-matrix inverse A^{-1}. We call it a scrambler because the map $x' = xA$ is a one-to-one permutation of all the k-dimensional R-vectors x. For example,

$$A = \begin{bmatrix} 1 & D+1 \\ 1 & D \end{bmatrix}$$

$$A^{-1} = \begin{bmatrix} D & D+1 \\ 1 & 1 \end{bmatrix}$$

are two inverse binary polynomial 2×2 scramblers. (The reader may at first be surprised, as was the author, by the existence of nontrivial pairs of scramblers that are feedback-free and thus not subject to infinite error propagation.) The only 1×1 scramblers are the trivial ones consisting of units of R.

Now we illustrate an invariant factor decomposition of G with respect to R by the block diagram of Fig. 5. Input sequences are scrambled in the $k \times k$ R-scrambler A; the outputs are then operated on individually by the invariant factors γ_i; finally, the k outputs plus $n - k$ dummy zeroes are scrambled in an $n \times n$ R-scrambler B to give the output sequences.

The invariant-factor theorem and its extension are well known in linear system theory, particularly in the work of Kalman [11], [12], who attributes the first engineering use of the extension above to McMillan [13]. As far as the author is aware, its use has generally been confined to the rings of polynomials and rational functions. The utility of considering additional rings will become clear in the sections to follow.

III. Structural Theorems

Our principal results are presented in three sections. In the first, we show how to determine whether G has inverses of various kinds. In the second, we show that every encoder G is equivalent to a so-called minimal encoder, which is a conventional convolutional encoder with a feedback-free inverse. In the final section we point out other desirable properties of minimal encoders: they require as few memory elements to realize as any equivalent encoder, they allow easy enumeration of error events by length, and they ensure that short codewords correspond to short information sequences in a way not shared by nonminimal encoders. We conclude that a minimal encoder is the natural choice to generate any desired code. In two appendices, we discuss dual codes and systematic encoders; the latter may also be taken as canonical encoders for any code.

Inverses

In this section we shall determine when a $k \times n$ Q-matrix G has an R-matrix right inverse G^{-1}, where Q is a ring of fractions of R and $k \leq n$. The results are stated in terms of the invariant factors γ_i of G with respect to R. We assume $\gamma_k \neq 0$, otherwise G has rank less than k and thus no inverse of any kind.

Consider the invariant-factor decomposition of G with respect to R, as illustrated in Fig. 5. The outputs of the scrambler A when its inputs range over all possible sequences are simply all possible sequences in some different

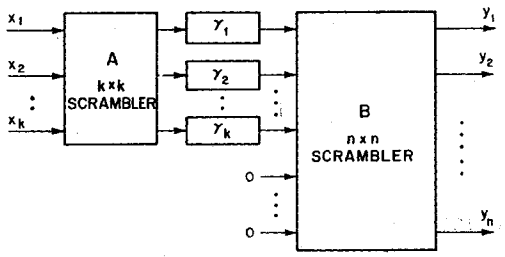


Fig. 5. Invariant-factor decomposition of (n, k) encoder.

order, since A is invertible. Moreover, if the inputs to A range over all vectors of k elements of R, then the outputs are all vectors of k elements of R in a different order, since A and A^{-1} are R-matrices. In particular, there is some input R-vector x_k to A such that the output x_kA is ε_k, the kth unit input, namely, $x_k = \varepsilon_kA^{-1}$. Now the input vector $\gamma_k^{-1}x_k$ gives $\gamma_k^{-1}\varepsilon_k$ at the output of A, and ε_k at the output of Γ, hence the R-vector ε_kB at the matrix output. But $\gamma_k^{-1}\varepsilon_k$, hence $\gamma_k^{-1}x_k$, is an R-vector if and only if γ_k^{-1} is an element of R. Continuing with this argument, we have the following.

Lemma 1: Let R be a principal ideal domain and Q a ring of fractions of R. Let G be a Q-matrix with invariant-factor decomposition $G = A\Gamma B$ with respect to R, and invariant factors $\gamma_i = \alpha_i/\beta_i$, $1 \leq i \leq k$. If $\alpha_k \neq 1$, then there is a vector $\gamma_k^{-1}\varepsilon_kA^{-1}$ that is not an R-vector but which gives an R-vector output.

Proof: If $\alpha_k \neq 1$, then $\gamma_k^{-1} = \beta_k/\alpha_k$ is not an element of R, hence $\gamma_k^{-1}\varepsilon_k$ is not an R-vector, hence $\gamma_k^{-1}\varepsilon_kA^{-1}$ is not an R-vector, since if it were $(\gamma_k^{-1}\varepsilon_kA^{-1})$ A would be an R-vector. But $\gamma_k^{-1}\varepsilon_kA^{-1}G = \gamma_k^{-1}\varepsilon_k\Gamma B = \varepsilon_kB$ is an R-vector since ε_k and B are in R. Q.E.D.

We call the numerator α_k of the kth invariant factor that appears in Lemma 1 the minimum factor of G with respect to R; this designation will be justified by Theorem 2.

From Lemma 1 we obtain a general theorem on inverses, application of which to particular rings R will settle many questions concerning inverses. We note that if $G = A\Gamma B$, then $G^{-1} = B^{-1}\Gamma^{-1}A^{-1}$ is an inverse for G, where Γ^{-1} is the $n \times k$ matrix with diagonal elements γ_i^{-1}. Since A^{-1} and B^{-1} are R-matrices, G^{-1} is certainly an R-matrix if all γ_i^{-1} are elements of R. The following theorem says that if some γ_i^{-1} is not an element of R, then G has no R-matrix inverse.

Theorem 1: Let R be a principal ideal domain and Q a ring of fractions of R. Let G be a Q-matrix whose invariant factors with respect to R are $\gamma_i = \alpha_i/\beta_i$, $1 \leq i \leq k$. Then the following statements are equivalent.

1) G has an inverse G^{-1}, which is an R-matrix.
2) There is no x that is not an R-vector such that $y = xG$ is an R-vector, or $y \in R$-implies $x \in R$.
3) $\alpha_k = 1$.

Proof: We shall show $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 1$.

$(1 \Rightarrow 2)$. If $y = xG$ is an R-vector, then $x = yG^{-1}$ is an R-vector since G^{-1} is an R-matrix.

$(2 \Rightarrow 3)$. By Lemma 1, if $\alpha_k \neq 1$, then $x = \gamma_k^{-1}\varepsilon_kA^{-1}$ is a vector not an R-vector such that xG is an R-vector.

$(3 \Rightarrow 1)$. If $\alpha_k = 1$, then $\alpha_i = 1$, $1 \leq i \leq k$, since $\alpha_i \mid \alpha_k$. Hence the inverse invariant factors $\gamma_i^{-1} = \beta_i/\alpha_i = \beta_i$ are all elements of R. If $G = A\Gamma B$ is an invariant-factor decomposition of G with respect to R, then A^{-1} and B^{-1} are R-matrices, so that $G^{-1} = B^{-1}\Gamma^{-1}A^{-1}$ is an R-matrix that serves as the desired inverse. Q.E.D.

We then obtain the results we need as special cases.

Corollary 1: An encoder G has a feedback-free inverse iff its minimum factor with respect to the polynomials $F[D]$ is 1.

Corollary 2: An encoder G has a feedback-free pseudo-inverse iff its minimum factor with respect to the finite sequences $F_{fl}(D)$ is 1. Furthermore, in this case and only in this case is there no infinite x such that $y = xG$ is finite.

Corollary 3: An encoder G has a realizable inverse iff its minimum factor with respect to the realizable functions $F_{rz}(D)$ is 1.

Corollary 4: An encoder G has a realizable pseudo-inverse iff its minimum factor with respect to the rational functions $F(D)$ is 1; that is, $\alpha_k \neq 0$, or the rank of G is k.

Here we have used the obvious facts that a feedback-free pseudo-inverse implies and is implied by a finite-sequence inverse, and similarly with a realizable pseudo-inverse and a rational inverse, where one is obtained from the other in both cases by multiplication by $D^{\pm d}$, d being the delay of the pseudo-inverse. We also note that since G is both an $F_{rz}(D)$ and an $F(D)$ matrix, the invariant factors with respect to these rings cannot have denominator terms; further, since $F(D)$ is a field, the only greatest common divisors are 1 and 0, and the rank of G equals the rank of Γ since A and B are invertible.

With equal ease, we can obtain a sharper result on pseudo-inverses $\tilde{G}^{-1}$ in the cases where G has no inverse. We make the following general definition of a pseudo-inverse.

Definition: $\tilde{G}^{-1}$ is an R-matrix pseudo-inverse for G with factor ψ if $\tilde{G}^{-1}$ is an R-matrix and $G\tilde{G}^{-1} = \psi I_k$.

If $G^{-1} = B^{-1}\Gamma^{-1}A^{-1}$ is not an R-matrix inverse for G, it is because Γ^{-1} is not an R-matrix. Since $\gamma_i^{-1} = \beta_i/\alpha_i$ and $\alpha_i \mid \alpha_k$, $1 \leq i \leq k$, $\tilde{G}^{-1} = \alpha_kG^{-1} = B^{-1}(\alpha_k\Gamma^{-1})A^{-1}$ is an R-matrix pseudo-inverse for G with factor α_k. Theorem 2 shows that this is the minimum such factor.

Theorem 2: Let R be a principal ideal domain and Q a ring of fractions of R. Let G be a Q-matrix whose invariant factors with respect to R are $\gamma_i = \alpha_i/\beta_i$, $1 \leq i \leq k$. Then G has an R-matrix pseudo-inverse $\tilde{G}^{-1}$ with factor α_k; further, all R-matrix pseudo-inverses have factors ψ such that α_k divides ψ.

Proof: The discussion preceding the theorem shows how to construct a pseudo-inverse with factor α_k. Therefore let $\tilde{G}^{-1}$ be any R-matrix pseudo-inverse, and consider the input $x = \gamma_k^{-1}\varepsilon_kA^{-1}$. By Lemma 1, xG is an R-vector, hence $xG\tilde{G}^{-1}$ is an R-vector, but

$$xG\tilde{G}^{-1} = \psi x$$
$$= \psi\gamma_k^{-1}\varepsilon_kA^{-1}.$$

Hence $(\psi\gamma_k^{-1}\epsilon_k A^{-1})A = \psi\gamma_k^{-1}\epsilon_k$ is an R-vector, which is to say $\psi\gamma_k^{-1} = \psi\beta_k/\alpha_k$ is an element of R. But gcd $(\alpha_k, \beta_k) = 1$; hence α_k must divide ψ in R. Q.E.D.

We note that we could have obtained Theorem 1 as a consequence of Theorem 2 and Lemma 1.

The pseudo-inverses we are interested in are realizable pseudo-inverses with delay d, or, in the terminology introduced above, $F_{rz}(D)$-matrix pseudo-inverses with factor D^d. Since the only prime in the ring of realizable functions $F_{rz}(D)$ is D, and since G is itself realizable, the invariant factors of G with respect to $F_{rz}(D)$ are $\gamma_i = D^{d_i}$ (or zero); and in particular the minimum factor is D^{d_k}. We then define the delay d of a realizable matrix as $d = d_k$, so $\alpha_k = D^d$ and $d_i \leq d$, $1 \leq i \leq k$. Then Theorem 2 answers a problem stated by Kalman ([11], 10.10e) with the following corollary.

Corollary 1: Let G be realizable and let the minimum factor with respect to the realizable functions $F_{rz}(D)$ be $\alpha_k = D^d$. Then G has a realizable pseudo-inverse $\tilde{G}^{-1}$ with delay d, and no realizable pseudo-inverse with delay less than d.

Similarly, the question of the delay of a feedback-free inverse, which was investigated in [4] and [7], is answered by Corollary 2.

Corollary 2: Let G be realizable and let the minimum factor with respect to the polynomials $F[D]$ be α_k. Then G has a feedback-free pseudo-inverse $\tilde{G}^{-1}$ with delay d' if and only if $\alpha_k = D^d$ for $d' \geq d$.

For computation, it is convenient not to have to compute invariant factors repeatedly, so we use the following lemma.

Lemma 2: Let G have invariant factors γ_i with respect to R; then the invariant factors with respect to Q are γ_i', where $\gamma_i' = \gamma_i$ up to units in Q and γ_i' is a product of primes in Q.

Proof: Let $\gamma_i = \gamma_i'\,\gamma_i''$, with γ_i'' a unit in Q. Let $G = A\Gamma B$ be an invariant factor decomposition of G with respect to R. Let B' be the Q matrix obtained from B by multiplying the ith row by γ_i''; then det $B' = (\det B)\,\pi_i\gamma_i''$ is a unit in Q, since det B is a unit in R. Hence $G = A\Gamma' B'$ is an invariant-factor decomposition of G with respect to Q, so the γ_i' are invariant factors of G with respect to Q. Q.E.D.

Now we have the following recipe for deciding whether G has inverses of various types. Let $G = \{h_i/\psi_i\}$; multiply each row through by its denominator to obtain the polynomial matrix $G' = \{h_i\}$. Compute all $k \times k$ subdeterminants, and find their greatest common divisor θ_k. Let $\Delta_k = \theta_k/\pi_i\psi_i$ and reduce to lowest terms. Now $\Delta_k = \pi_i\gamma_i = \pi\alpha_i/\pi\beta_i = A_k/B_k$. If $A_k = 1$, G has a polynomial inverse. If $A_k = D^d$, G has a polynomial pseudo-inverse. If A_k is a polynomial not divisible by D, G has a realizable inverse. Finally, if $A_k \neq 0$, G has a realizable pseudo-inverse. The minimum pseudo-inverse delay d is the greatest common delay of the $k \times k$ subdeterminants minus the greatest common delay of the $(k-1) \times (k-1)$ subdeterminants.

In [7], Olson gives a test for the existence and minimum delay of any feedforward inverse that is equivalent to the above; although more cumbersome, Olson's result and proof are remarkable for being carried through successfully without the aid of the powerful algebraic tools used here.

Canonical Encoders Under Equivalence

The theorems of this section are aimed at the determination of a canonical encoder in the equivalence class of encoders generating a given code.

The idea of the first theorem of this section is as follows. Any encoder G has the invariant factor decomposition $A\Gamma B$ with respect to the polynomials $F[D]$ as is illustrated in Fig. 5, where A is a $k \times k$ polynomial scrambler, B is an $n \times n$ polynomial scrambler, and Γ is a set of generally nonpolynomial transfer functions. If the inputs to A are all the k-tuples of sequences; then since A is invertible the outputs are also all the k-tuples in some different order. If none of the γ_i is zero, then the outputs of Γ are also all k-tuples of sequences, if we allow many-to-one encoders, so that G may have rank $r < k$ and $\gamma_{r+1} = \cdots = \gamma_k$ may be zero, then the outputs of Γ are all k-tuples of sequences in which the last $k - r$ components are zero. But the outputs of B with these inputs are the code generated by G; G is therefore equivalent to the encoder G_0 represented by the first r rows of B. Now since B is polynomial and has a polynomial inverse, G_0 is also polynomial and has a polynomial right inverse G_0^{-1} consisting of the first r columns of B^{-1}. These observations are made precise in the following theorem and proof.

Theorem 3: Every encoder G is equivalent to a conventional convolutional encoder G_0 that has a feedback-free delay-free inverse G_0^{-1}.

Remark: In other words, G_0 and G_0^{-1} are polynomial and $G_0G_0^{-1} = I_r$.

Proof: Let G have invariant-factor decomposition $G = A\Gamma B$ with respect to $F[D]$. Let G_0 be the first r rows of B,

$$G_0 = \{b_i\} \qquad 1 \leq i \leq r.$$

G_0 is polynomial since B is, and has a polynomial inverse G_0^{-1} equal to the first r columns of B^{-1}. To show equivalence, let y_0 be any codeword in the code generated by G; then

$$\begin{aligned} y_0 &= x_0 G \\ &= (x_0 A\Gamma)(B) \\ &= \sum_{i=1}^{r} (x_0 A\Gamma)_i b_i \\ &= x_1 G_0, \end{aligned}$$

where x_1 is the vector consisting of the first r sequences of $x_0 A\Gamma$; hence any codeword in the code generated by G is also in the code generated by G_0. Conversely, let y_1 be any word in the code generated by G_0; then

$$\begin{aligned} \boldsymbol{y}_1 &= \boldsymbol{x}_1 G_0 \\ &= \boldsymbol{x}_1' B \\ &= (\boldsymbol{x}_0' A \Gamma) B \\ &= \boldsymbol{x}_0' G, \end{aligned}$$

where $\boldsymbol{x}_1'$ is the k-dimensional vector equal to $\boldsymbol{x}_1$ in the first r positions and to zero thereafter, while $\boldsymbol{x}_0' = \boldsymbol{x}_1' \Gamma^{-1} A^{-1}$, where Γ^{-1} has diagonal elements γ_i^{-1} for $\gamma_i \neq 0$ and 0 for $\gamma_i = 0$; hence any codeword in the code generated by G_0 is also in the code generated by G. Q.E.D.

At this point we remark that all (n, n) encoders are obviously uninteresting: for if G has full rank, $\gamma_n \neq 0$, then the code consists of all n-tuples of sequences, hence G is equivalent to the identity encoder I_n; while if $\gamma_n = 0$, then G has rank $r < n$ and is equivalent to some (n, r) code.

Let us define any encoder meeting the conditions on G_0 in Theorem 3 as basic (from the fact that the set of generators in such an encoder is a basis for the $F[D]$-module of polynomial codewords).

Definition 4: A *basic* encoder G is a conventional convolutional encoder with a feedback-free inverse G^{-1}; that is, G is polynomial, G^{-1} is polynomial, and $GG^{-1} = I_k$.

In virtue of Corollary 1 to Theorem 1, an encoder is basic if and only if it is polynomial and has minimum factor with respect to $F[D]$ of 1, so that all invariant factors are equal to 1.

Basic encoders are not in general unique; the following theorem characterizes the class of basic encoders generating any code.

Theorem 4: If G is a basic encoder, G' is an equivalent basic encoder if and only if $G' = TG$, where T is a $k \times k$ polynomial scrambler; that is, T is a square polynomial matrix with polynomial inverse T^{-1}.

Proof: Let $G' = TG$; then both G' and $G'^{-1} = G^{-1}T^{-1}$ are polynomial since T, G, G^{-1}, and T^{-1} are all polynomial, so G' is basic. G' is equivalent to G since if $\boldsymbol{y}_0 = \boldsymbol{x}_0 G$, then $\boldsymbol{y}_0 = \boldsymbol{x}_1 G'$ where $\boldsymbol{x}_1 = \boldsymbol{x}_0 T^{-1}$, while if $\boldsymbol{y}_0 = \boldsymbol{x}_0 G'$, then $\boldsymbol{y}_0 = \boldsymbol{x}_1 G$ where $\boldsymbol{x}_1 = \boldsymbol{x}_0 T$. Conversely, let G' be a basic encoder equivalent to G; then the generators $\boldsymbol{g}_i$ of G are in both codes, being generated by $\boldsymbol{\varepsilon}_i$ in the latter case and by input vectors

$$\boldsymbol{t}_i = \boldsymbol{g}_i G'^{-1}$$

in the former, since $\boldsymbol{t}_i G' = \boldsymbol{g}_i$. The $\boldsymbol{t}_i$ are polynomial since $\boldsymbol{g}_i$ and G'^{-1} are. Let T then be the matrix that has the $\boldsymbol{t}_i$ for rows; then $G = TG'$. We can repeat the argument with G and G' reversed to obtain $G' = SG$ for some polynomial matrix S. Since both G and G' have full rank, T and S must be invertible, and since $G = TSG$, S must be the inverse of T and vice versa. Q.E.D.

In the special case $k = 1$ (a rate-$1/n$ code), the basic encoder generating any code is uniquely defined up to units (and in the binary case, uniquely defined period, since the only unit is 1), since the only 1×1 scramblers are the units of $F[D]$, or the nonzero elements of F. Given any $(n, 1)$ code with generator $\boldsymbol{g} = \boldsymbol{h}/\psi$, where $\boldsymbol{h}$ is a vector of n polynomials, we can find the equivalent basic encoder by multiplying through by the denominator ψ and then canceling the greatest common divisor of the elements of $\boldsymbol{h}$. These facts have been known for some time [4], and represent all that this paper has to say about rate-$1/n$ codes. Since it is generally expected that there are good codes in the $(n, 1)$ class, in fact the best codes for most practical purposes, it is questionable whether the results of this paper will assist in the search for constructive methods of obtaining good codes.

For (n, k) encoders with $k \geq 2$, we can distinguish a subclass of basic encoders with further useful properties. By appealing to realizability theory (see Appendix II), one can show that a basic encoder G that has maximum degree μ among its $k \times k$ subdeterminants can be realized with μ and no fewer than μ memory elements. However, the obvious realization in general requires $\Sigma \nu_i = \nu$ memory elements. We therefore define the following.

Definition 5: A basic encoder G is *minimal* if its overall constraint length ν in the obvious realization is equal to the maximum degree μ of its $k \times k$ subdeterminants.

We note first that μ is invariant over all equivalent basic encoders; for if G and G' are two such encoders, by Theorem 4 $G = TG'$ where $\det T$ is a unit in $F[D]$, so that the $k \times k$ subdeterminants of G are those of G' up to units. We now proceed to show that among all equivalent basic encoders there is at least one that is minimal. It is helpful to consider the backwards encoder $\boldsymbol{G}$ associated with any basic encoder, defined as the encoder with generators $D^{-\nu_i}\boldsymbol{g}_i$. As thus defined, $\boldsymbol{G}$ has elements that are polynomials in D^{-1}, hence is anticausal rather than causal. Another way of stating this definition is to let V be the $k \times k$ diagonal matrix with diagonal elements $D^{-\nu_i}$; then

$$\boldsymbol{G} = V G.$$

We are interested in $\boldsymbol{G}$ because of the following lemma.

Lemma 3: If G is basic, G is minimal if and only if $\boldsymbol{G}$ has an anticausal inverse.

Remark: Recall that since we allow only sequences that "start" at some time, all anticausal sequences are actually polynomials in D^{-1} (elements of $F[D^{-1}]$), with finite negative delay and degree less than or equal to 0. Hence an anticausal inverse for $\boldsymbol{G}$ must in fact be an $F[D^{-1}]$-inverse.

Proof: If $\boldsymbol{G}$ has an anticausal inverse, then $G = I_k V^{-1} \boldsymbol{G}$ is (loosely speaking) an invariant-factor decomposition for G with respect to $F[D^{-1}]$, so that G has invariant factors $\gamma_i = (1/D^{-\nu_i})$ with respect to $F[D^{-1}]$ and $\pi\gamma_i = (1/D^{-\nu})$. From the extended invariant factor theorem, $\pi\gamma_i = \theta_k/\psi_k$, where θ_k is the greatest common divisor of the numerators and ψ_k the least common multiple of the divisors of the $k \times k$ subdeterminants of G as ratios of polynomials in D^{-1}. But since G is basic, these subdeterminants are polynomials in D with no common prime divisor, all of which can be expressed as ratios of polynomials in D^{-1} as $\Delta = \Delta'/D^{\delta}$, where Δ' is a polynomial

in D^{-1} and where δ is the degree of Δ as a polynomial in D. Hence $\theta_k = 1$ and $\psi_k = D^{-\mu}$, where μ is the maximum degree of any $k \times k$ subdeterminant of G. Hence $D^{\nu} = \theta_k/\psi_k = D^{\mu}$, so $\nu = \mu$ and G is minimal.

Conversely, if G is minimal, then at least one of the $k \times k$ subdeterminants of $\boldsymbol{G}$ has a nonzero constant term, since the $k \times k$ subdeterminants of $\boldsymbol{G}$ are those of G times the determinant of V, which is $D^{-\nu} = D^{-\mu}$, and at least one of the subdeterminants of G has degree μ. Thus D^{-1} is not a common factor of the $k \times k$ subdeterminants of $\boldsymbol{G}$; further, there can be no other common factor since such a factor would imply a common factor for G, but G is basic. Hence the $k \times k$ subdeterminants of $\boldsymbol{G}$ are relatively prime as polynomials in D^{-1}, so that $\boldsymbol{G}$ has invariant factors equal to 1 with respect to $F[D^{-1}]$ by the invariant-factor theorem, and consequently an anticausal inverse by Theorem 1. Q.E.D.

The desired result is then obtained with a constructive proof.

Theorem 5: Every encoder G is equivalent to a minimal encoder.

Proof: From Theorem 3 it is sufficient to prove the theorem when G is basic. We shall show that whenever G is basic but not minimal, there exists an equivalent basic encoder with reduced constraint length. By Lemma 3, $\boldsymbol{G}$, the backwards encoder associated with G, has no anticausal inverse, which in turn implies a factor of D^{-1} in all $k \times k$ subdeterminants of $\boldsymbol{G}$. In other words, the matrix $\boldsymbol{G}$ modulo D^{-1} (consisting of all constant coefficients of elements of $\boldsymbol{G}$) is singular. (Since these are the high-order (ν_ith-order) coefficients of G, the high-order coefficients equally form a singular matrix.) There is therefore a linear combination of the generators of $\boldsymbol{G}$ with coefficients in F that is equal to a vector divisible by D^{-1}, thus with all-zero constant terms; that is, for some nonzero vector $\boldsymbol{f}$ with elements in F,

$$\boldsymbol{f}\boldsymbol{G} = 0 \bmod D^{-1}.$$

It follows that

$$\boldsymbol{f}\boldsymbol{V}\boldsymbol{G} = 0 \bmod D^{-1};$$

or deg $(\boldsymbol{f}\boldsymbol{V}\boldsymbol{G}) < 0$, or consequently deg $(\boldsymbol{f}\boldsymbol{V}\boldsymbol{G}D^{\nu_0}) < \nu_0$, where ν_0 is any integer. Let us choose ν_0 equal to the largest constraint length of any generator $\boldsymbol{g}_i$ such that f_i is nonzero; then $\boldsymbol{x} = \boldsymbol{f}\boldsymbol{V}D^{\nu_0}$ is polynomial (in D), so $\boldsymbol{y} = \boldsymbol{x}\boldsymbol{G}$ is polynomial, but deg $\boldsymbol{y} < \nu_0 =$ deg $\boldsymbol{g}_i$ for some $\boldsymbol{g}_i$ upon which $\boldsymbol{y}$ is linearly dependent. Hence we can replace $\boldsymbol{g}_i$ by $\boldsymbol{y}$ to get an equivalent generator matrix with shorter constraint length. Q.E.D.

If we are actually interested in computing a minimal encoder equivalent to some given encoder G, we can proceed as follows. First we multiply through by denominator terms to obtain a polynomial matrix, and compute $k \times k$ subdeterminants. If ψ is some polynomial common to all such subdeterminants, then G mod ψ is singular since all $k \times k$ subdeterminants are zero mod ψ. Thus there is some linear combination of generators that equals zero mod ψ, which combination can be determined by inspection or systematically by reduction to triangular form, mod ψ. This same linear combination of generators not mod ψ will give a polynomial vector divisible by ψ; we cancel the ψ and use the result to replace one of the generators entering into the linear combination. Thus we will eventually arrive at a basic encoder. If the basic encoder is not minimal, there is some linear combination of the generators of the associated backwards encoder that is equal to 0 mod D^{-1}, which can be used to replace one of the backwards-encoder generators. Since the mod D^{-1} leaves only the zero-order terms of the backward generators, we can work instead with the ν_ith-order terms of the basic encoder, and continue until the matrix of high-order coefficients is nonsingular over $\boldsymbol{F}$, when we will have our minimal encoder.

Properties of Minimal Encoders

The most important property of minimal encoders is that they are also minimal in the sense of requiring as few memory elements as any equivalent encoder. They also have a property, called the predictable degree property, useful in analyzing error events.

The proof of the former result depends on state-space arguments. We recall our definition of the projection operators P and $Q = 1 - P$ as the operators that truncate to times ≤ 0 and ≥ 1, respectively, and the abstract definition of the states as the set of outputs at time 1 and later for inputs zero at time 1 and later; for any input $\boldsymbol{x}$ the associated state is

$$\boldsymbol{s} = \boldsymbol{x}PGQ.$$

We shall show that the state space Σ_m of a minimal encoder G_m has no greater dimension than the state space Σ_G of any other equivalent encoder G. The result will then follow from the following lemma.

Lemma 4: If an encoder G can be realized with ν memory elements, dim $\Sigma_G \geq \nu$. Equality holds for minimal encoders in the obvious realization.

Proof: If F has q elements, there are q^{ν} physical states, and $q^{\dim \Sigma G}$ abstract states. Since G is causal the output at time 1 and later when the inputs stop at time 0 is uniquely defined by the physical state. Hence the number of physical states is not less than the number of abstract states.

For minimal encoders in the obvious realization, the physical states correspond to the q^{ν} states

$$\boldsymbol{s} = \left[\sum_{i=1}^{k}\sum_{j=0}^{\nu_i-1} f_{ij}D^{-j}\boldsymbol{g}_i\right]Q \qquad f_{ij} \in F.$$

The claim is that all such states are different, or, in view of linearity, that no state is equal to zero unless all $f_{ij} = 0$. Consider the input $\boldsymbol{x}$ to the backwards encoder $\boldsymbol{G}$ defined by

$$\boldsymbol{x} = \sum_{i=1}^{k}\sum_{j=0}^{\nu_i-1} f_{ij}D^{\nu_i-j}\boldsymbol{\varepsilon}_i;$$

if any $f_{ij} \neq 0$ then deg $\boldsymbol{x} > 0$. The corresponding output is

$$y = xG$$
$$= \sum_{i=1}^{k} \sum_{j=0}^{\nu_i - 1} f_{ij} D^{-j} g_i,$$

which gives one of the states $s = yQ$ defined above. Now if $s = 0$, then deg $y \leq 0$, which is impossible since G has an anticausal inverse, so deg $y \leq 0$ implies deg $x \leq 0$ from Theorem 1. Hence the physical state space and the abstract state space are isomorphic. Q.E.D.

Realizability theory (see Appendix II) shows that any linear circuit can actually be realized with dim Σ_G memory elements; such a realization of a particular G is called canonical in linear system theory.

In the present context, where we are concerned with equivalence classes of encoders G, it seems appropriate to call an encoder canonical if the dimension of its state space is minimum over all equivalent encoders. The next lemma shows that canonical encoders are those with polynomial inverses, which is curiously close to the feedback-free inverse condition that eliminates catastrophic error propagation. The idea of the lemma is to look at the set CQ of all codewords truncated to $t \geq 1$. Some of these truncated codewords are still equal to codewords, namely, the codewords in the set C_{Gm} that do not actually "start" until time 1 or later. The remainder must be equal to codewords plus a state, regardless of the encoder that generates the code. Each of the equivalence classes of CQ modulo C_{Gm} must therefore contain at least one state. But such equivalence classes S_{Gm} form a vector space of dimension dim S_{Gm} so that the minimum state-space dimension is dim S_{Gm}. The remainder of the proof shows that Σ_G is isomorphic to S_{Gm} if and only if G has a polynomial inverse.

Lemma 5: Let Σ_m be the state space of a minimal encoder G_m, and Σ_G the state space of any equivalent encoder G. Then dim $\Sigma_m \leq$ dim Σ_G, with equality if and only if G has a polynomial inverse.

Proof: Let CQ be the space of all yQ, y any codeword in C. Let C_{Gm} and C_G be the spaces of all $y = xG_m$ or xG such that del $x \geq 1$. It is easy to verify that these are all vector spaces over F, and to see that C_{Gm} and C_G are subspaces of CQ. Now by Theorem 1 with $R = F[[D]]$, the ring of formal power series in D, del $y \geq 0$ implies del $x \geq 0$ if and only if G has a causal inverse, which by constancy is the same as del $y \geq 1$ implies del $x \geq 1$. Since G_m has a causal inverse, C_{Gm} consists of all y with del $y \geq 1$. All elements in C_G are therefore in C_{Gm}, with equality iff C_G has a causal (hence realizable) inverse.

Let $S_G = CQ/C_G$ be the equivalence classes of CQ modulo C_G; that is, for any $yQ \in CQ$, an element in S_G is the set of all $y'Q$ such that

$$yQ = y'Q + xG \qquad \text{del } x \geq 1.$$

From its definition, it is immediate that S_G is a vector space over F, a subspace of CQ, and that CQ has the direct sum decomposition

$$CQ = C_G \oplus S_G.$$

Similarly, we can define $S_{Gm} = CQ/C_{Gm}$, and obtain

$$CQ = C_{Gm} \oplus S_{Gm}.$$

Since $C_G \leq C_{Gm}$, $S_{Gm} \leq S_G$.

Now each such equivalence class contains a state. For any yQ can be expressed as

$$yQ = xGQ$$
$$= xPGQ + xQGQ$$
$$= s + x'G \qquad s \text{ a state, del } x' \geq 1,$$

so yQ is equivalent to s modulo C_G. Hence $S_G \leq \Sigma_G$ in the sense of isomorphism, and $S_{Gm} \leq \Sigma_m$. Equality holds if each equivalence class contains only one state, so that if s_1 and s_2 are any different states,

$$s_1 \not\equiv s_2 \bmod C_G.$$

If s is the state $s_1 - s_2$, this means

$$s \neq y = xG \qquad \text{del } x \geq 1.$$

Let $s = x_1GQ$ for some x_1 with deg $x_1 \leq 0$; then

$$s - y = x_2GQ \neq 0,$$

where $x_2 = x_1 - x$, so deg $x_2 \geq 1$. Now deg $x_2G \leq 1$ implies deg $x_2 \leq 0$ for both x_2g and x_2 finite if and only if G has an $F[D^{-1}]$inverse, by Theorem 1 with $R = F[D^{-1}]$. The backwards encoder $\mathbf{G}$ corresponding to any minimal encoder has such an inverse, hence a fortiori so does G, so $S_{Gm} = \Sigma_m$. In summary, in the sense of isomorphism,

$$\Sigma_m = S_{Gm} \leq S_G \leq \Sigma_G,$$

where the second equality holds iff G has a causal inverse, and the third iff G has an $F[D^{-1}]$ inverse. Since the invariant factors of G with respect to $F[D^{-1}]$ are the same as those with respect to $F[D]$ except for powers of D, and the invariant factors of $G = A\Gamma B$ with respect to $F[D^{-1}]$ do not contain negative powers of D, since they are realizable, we see that the condition that G have a polynomial inverse is necessary and sufficient for equality. Q.E.D.

In view of Lemma 4, we now have the following required theorem.

Theorem 6: Let G be an encoder realizable with ν memory elements and let G_m be an equivalent minimal encoder with overall constraint length μ, hence μ memory elements in the obvious realization. Then $\nu \geq \mu$.

Proof:

$$\nu \geq \dim \Sigma_G \geq \dim \Sigma_m = \mu. \qquad \text{Q.E.D.}$$

From Lemma 5, the class of encoders that can be realized with dim Σ_m memory elements is quite broad, and includes basic encoders. Minimal encoders are unique, however, in being realizable with dim Σ_m memory elements as conventional encoders in the obvious realization. It is of course possible for a nonminimal encoder to be less complex than an equivalent minimal encoder by virtue of having fewer adders, multipliers, interconnections, and

so forth; furthermore, system complexity is usually dominated by the codeword estimator, not the encoder. However, Theorem 6 does tend to discourage spending much time on looking for great encoder simplifications through unconventional approaches, such as the use of feedback.

Another property of minimal encoders, called the predictable degree property, is a useful analytical tool. Note that in general for any conventional encoder with constraint lengths ν_i, and any codeword $\boldsymbol{y} = \boldsymbol{x}G$,

$$\begin{aligned}\deg \boldsymbol{y} &\leq \max_{1\leq i\leq k} (\deg x_i \boldsymbol{g}_i) \\ &= \max_{1\leq i\leq k} (\deg x_i + \nu_i).\end{aligned}$$

If equality holds for all $\boldsymbol{x}$, we say G has the predictable degree property. Now we have the following.

Lemma 6: Let G be a basic encoder; then G has the predictable degree property if and only if G is minimal.

Proof: By Lemma 3, G is minimal iff its backwards encoder $\boldsymbol{G}$ has an anticausal inverse. By Theorem 1 with $R = F[D^{-1}]$, $\boldsymbol{G}$ has an anticausal inverse iff $\deg \boldsymbol{x}\boldsymbol{G} \leq 0$ implies $\deg \boldsymbol{x} \leq 0$, or equivalently iff $\deg \boldsymbol{x} \geq 1$ implies $\deg \boldsymbol{x}\boldsymbol{G} \geq 1$, or by constancy $\deg \boldsymbol{x} \geq d$ implies $\deg \boldsymbol{x}\boldsymbol{G} \geq d$. But

$$\begin{aligned}\boldsymbol{y} &= \boldsymbol{x}\boldsymbol{G} \\ &= \sum_{i=1}^{k} x_i D^{-\nu_i} \boldsymbol{g}_i \\ &= \boldsymbol{x}'G\end{aligned}$$

where the $\boldsymbol{x}'$ corresponding to $\boldsymbol{x}$ is given by

$$\boldsymbol{x}' = \sum_{i=1}^{k} x_i D^{-\nu_i} \boldsymbol{\varepsilon}_i.$$

Now $\deg \boldsymbol{x} \geq d$ iff $\max \deg x_i \geq d$ iff $\max (\deg x_i' + \nu_i) \geq d$. Hence $\deg \boldsymbol{x} \geq d$ implies $\deg \boldsymbol{x}\boldsymbol{G} \geq d$ iff $\max (\deg x_i' + \nu_i) \geq d$ implies $\deg \boldsymbol{y} = \deg \boldsymbol{x}'G \geq d$, which is the same as the predictable degree property. Q.E.D.

In our earlier discussion, we asserted that the error events of interest are the finite codewords. Let us normalize all such words $\boldsymbol{y}$ to start at time zero, $\operatorname{del} \boldsymbol{y} = 0$. When G has a zero-delay feedback-free inverse, these are precisely the words generated by inputs $\boldsymbol{x}$ that are finite and start at zero, $\operatorname{del} \boldsymbol{x} = 0$. When G has the predictable degree property as well, we can easily enumerate the finite codewords by degree, since for each possible input $\boldsymbol{x}$ we can compute the degree of the output knowing only the constraint lengths ν_i. In fact the number of codewords of degree $\leq d$ is equal to the number of ways of choosing k polynomials x_i such that $\deg x_i \leq d - \nu_i$ or $x_i = 0$. For example, if an $(n, 2)$ binary code has $\nu_1 = 2$, $\nu_2 = 4$, then there is one codeword of degree less than 2 (the all-zero word); there are two of degree ≤ 2, 4 of degree ≤ 3, 16 of degree ≤ 4, 64 of degree ≤ 5, and so forth. Of course, all equivalent encoders have the same codewords and hence the same distribution of codeword lengths.

The predictable degree property also guarantees that in some sense short codewords will be associated with short information sequences. Let us establish a partial ordering of information sequences such that $\boldsymbol{x} < \boldsymbol{x}'$ if $\deg x_i \leq \deg x_i'$ for all i, with at least one strict inequality. Codewords $\boldsymbol{y}$ can be ordered by their degrees $\deg \boldsymbol{y}$, namely, the maximum degree of all components y_i. Now we have the following:

Lemma 7: $\boldsymbol{x} < \boldsymbol{x}'$ implies $\deg \boldsymbol{y} \leq \deg \boldsymbol{y}'$ if and only if G is minimal, where $\boldsymbol{y} = \boldsymbol{x}G$, $\boldsymbol{y}' = \boldsymbol{x}'G$.

Proof: If G is minimal, and $\boldsymbol{x} < \boldsymbol{x}'$, then

$$\begin{aligned}\deg \boldsymbol{y} &= \max (\deg x_i + \nu_i) \\ &\leq \max (\deg x_i' + \nu_i) \\ &= \deg \boldsymbol{y}'.\end{aligned}$$

If G is not polynomial, then it has an infinite generator $\boldsymbol{g}_i = \boldsymbol{h}_i/\psi_i$ where $\boldsymbol{h}_i$ and ψ_i are polynomial. Let $\boldsymbol{x} = \boldsymbol{\varepsilon}_i$, $\boldsymbol{x}' = \psi_i \boldsymbol{\varepsilon}_i$; then $\boldsymbol{x} < \boldsymbol{x}'$ but $\boldsymbol{y} = \boldsymbol{g}_i$ is infinite whereas $\boldsymbol{y}' = \boldsymbol{h}_i$ is finite, hence $\deg \boldsymbol{y}' < \deg \boldsymbol{y} = \infty$.

If G is polynomial but not basic, then by Lemma 1 there is an infinite input $\boldsymbol{x} = \gamma_k^{-1} \boldsymbol{\varepsilon}_k A^{-1}$ that gives a finite output $\boldsymbol{y}$, but $\boldsymbol{x}' = \beta_k \boldsymbol{\varepsilon}_k A^{-1}$ is finite and gives finite output $\boldsymbol{y}' = \alpha_k \boldsymbol{y}$ where $\gamma_k = \alpha_k/\beta_k$, $\deg \alpha_k \geq 1$. Hence $\boldsymbol{x}' < \boldsymbol{x}$, but $\deg \boldsymbol{y}' = \deg \alpha_k + \deg \boldsymbol{y} > \deg \boldsymbol{y}$.

If G is basic but does not have the predictable degree property, then for some $\boldsymbol{x}$

$$\deg \boldsymbol{y} < \max (\deg x_i + \nu_i).$$

Let $\boldsymbol{x}' = x_i \boldsymbol{\varepsilon}_i$ for some i for which the maximum on the right is attained; then $\boldsymbol{x}' < \boldsymbol{x}$, but

$$\deg \boldsymbol{y}' = \deg x_i + \nu_i > \deg \boldsymbol{y}.$$

Q.E.D.

This lemma is not quite as sharp as one would like, since the ordering of the inputs $\boldsymbol{x}$ is only partial, so that $\deg \boldsymbol{y} > \deg \boldsymbol{y}'$ does not imply $\boldsymbol{x} > \boldsymbol{x}'$ but only $\boldsymbol{x} \not< \boldsymbol{x}'$. Also, the ordering of codewords $\boldsymbol{y}$ by degree does not take into account the lengths of the individual sequences y_i. However, Lemma 7 reassures us that by choosing a minimal encoder we will not get an excessive number of information errors out per error event.

IV. Conclusions

All our results tend to the same conclusion: regardless of what convolutional code we want to use, we *can* and *should* use a minimal encoder to generate it. Minimal encoders are therefore to be considered a canonical class, like systematic encoders for block codes. (In Appendix II we consider systematic encoders as a canonical class for convolutional codes.)

It should be noted that none of our results depends on the finiteness of F, so that all apply to sampled-data filters, with F the field of real or complex numbers. (In Lemma 4 we do need some continuity restriction, such as that the output be a linear function of the state, when F is infinite; otherwise a multidimensional abstract state space could be mapped into a single real physical memory

element, for example, by a Cantor mapping.) In this context polynomial generators correspond to tapped delay line filters. The results on inverses are of clear interest here, but whether the remaining results are or not depends on whether our definition of equivalence is germane to some sampled-data problem.

There are several obvious directions for future work. First, the essential similarity in statement and proof of Theorems 3 and 5 suggests that there ought to be some way of setting up the problem so that the equivalence of any encoder to a minimal encoder could be shown without the intermediary of basic encoders. Second, there ought to be some way of treating the constraint lengths μ_i referenced to the output ($\mu_i = \max_{1 \le i \le k} \deg g_{ij}$) comparable in simplicity to our treatment of the constraint lengths ν_i referenced to the inputs. Third, at least on memoryless channels, permutations of the transmitted sequences or shifts of one relative to the others do not result in essentially different codes; it might be interesting to study encoders and codes under such a more general definition of equivalence (i.e., including column permutations of G and multiplication of columns by D^n). Finally, of course, the problem of constructing classes of "good" codes remains outstanding.

Appendix I

Dual Codes and Syndromes

We saw in Theorem 3 that all encoders were equivalent to an encoder G whose rows were the first k rows of a polynomial matrix B; G had a polynomial inverse G^{-1} that consisted of the first k columns of B^{-1}. It may have seemed that the last $n - k$ rows of B and columns of B^{-1} had no purpose; here we show how we may derive dual codes and form syndromes from them.

Dual Codes

Let B be an $n \times n$ polynomial matrix with polynomial inverse B^{-1}. Any k rows of B can be taken as the generators G of an (n, k) code; the remaining $n - k$ rows, which we call $(H^{-1})^T$, where T means transpose, can be taken as the generators of another $(n, n - k)$ code. G and $(H^{-1})^T$ have polynomial inverses G^{-1} and H^T formed from the corresponding columns of B^{-1}. Clearly if $\boldsymbol{y}$ is any codeword in the first code, say $\boldsymbol{y} = \boldsymbol{x}G$, then $\boldsymbol{y}H^T = 0$, since $\boldsymbol{y}$ is a linear combination of generators $\boldsymbol{g}_i$ all of which satisfy $\boldsymbol{g}_i H^T = 0$, by virtue of $\boldsymbol{g}_i$ being a row in B and H^T consisting of noncorresponding columns of B^{-1}. The code generated by G can be defined equally well as the set of row vectors $\boldsymbol{y}$ such that $\boldsymbol{y}H^T = 0$, or the null space of H^T.

The set of row vectors $\boldsymbol{z} = \boldsymbol{x}H$ for $\boldsymbol{x}$ any $(n - k)$-dimensional row vector of sequences can be considered to be a code generated by H, called the dual code, since if $\boldsymbol{y}$ is generated by G, $\boldsymbol{z}$ by H, then the inner product $\boldsymbol{y}\boldsymbol{z}^T$ is zero by virtue of $GH^T = 0$. The following lemma leads to an interesting theorem relating a code to its dual code.

Lemma 8: The $(n - k) \times (n - k)$ subdeterminants of H are equal up to units to the $k \times k$ subdeterminants of G.

Proof: Recall that G is the first k rows of B, while H^T is the last $(n - k)$ columns of B^{-1}, and $\det B =$ a unit in $F[D]$. We shall show that the upper left $k \times k$ subdeterminant of B is equal to the lower right $(n - k) \times (n - k)$ subdeterminant of B^{-1}; the same proof carries through for any other selection of columns in B and the corresponding rows in B^{-1} by transposition.[1] Write

$$B = \left[\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right] \qquad B^{-1} = \left[\begin{array}{c|c} B'_{11} & B'_{12} \\ \hline B'_{21} & B'_{22} \end{array}\right],$$

where the dotted lines separate rows and columns into groups of k and $n - k$. Consider the matrix product

$$\left[\begin{array}{c|c} B_{11} & B_{12} \\ \hline 0 & I_{n-k} \end{array}\right] \left[\begin{array}{c|c} B'_{11} & B'_{12} \\ \hline B'_{21} & B'_{22} \end{array}\right] = \left[\begin{array}{c|c} I_k & 0 \\ \hline B'_{21} & B'_{22} \end{array}\right];$$

taking determinants we have

$$|B_{11}|\,|B^{-1}| = |B'_{22}|,$$

but $|B^{-1}|$ is a unit in $F[D]$, hence $|B_{11}| = |B'_{22}|$ up to units. Q.E.D.

Now we have the following.

Theorem 7: If G is equivalent to a minimal encoder with overall constraint length μ, then the dual code can also be generated by a minimal encoder with overall constraint length μ.

Proof: Let $G = A\Gamma B$ be an invariant-factor decomposition of G with respect to the polynomials $F[D]$; the first k rows of B represent an equivalent basic encoder G'. The $k \times k$ subdeterminants of G' therefore have greatest common divisor 1. Furthermore, if μ is the maximum degree of any $k \times k$ subdeterminant of G', then by the discussion leading to Theorem 5 any minimal encoder equivalent to G' has overall constraint length μ. The transpose of the last $n - k$ columns of B^{-1} represents a polynomial encoder H for the code that is dual to that generated by G' and hence G. By Lemma 8, the $(n - k) \times (n - k)$ subdeterminants of H, being the same as those of G, have greatest common divisor equal to 1 and greatest degree μ. Hence H is basic, and is equivalent to a minimal encoder with overall constraint length μ. Q.E.D.

Syndromes

To use syndromes, the receiver must be able to make tentative individual symbol decisions on the elements of the codeword $\boldsymbol{y}$. These we call the received word $\boldsymbol{y}_r$; unlike the output $\hat{\mathbf{Y}}$ of a codeword estimator, $\boldsymbol{y}_r$ is not required to be a codeword. The received errors $\boldsymbol{e}_r$ are defined by

$$\boldsymbol{e}_r = \boldsymbol{y}_r - \boldsymbol{y},$$

where $\boldsymbol{y}$ is the codeword actually sent.

[1] We are indebted to Prof. R. G. Gallager for the main idea of the proof.

Let G be basic and equal to the first k rows of a polynomial invertible matrix B; the polynomial inverse B^{-1} is then made up of two parts, the inverse G^{-1} and the dual encoder transpose H^T. If we pass the received word $\boldsymbol{y}_r$ through B, the output may be correspondingly divided into two parts, the noisy estimates defined by

$$\begin{aligned}\boldsymbol{x}_r &= \boldsymbol{y}_r G^{-1} \\ &= (\boldsymbol{y} + \boldsymbol{e}_r)G^{-1} \\ &= \boldsymbol{x} + \boldsymbol{e}_r G^{-1},\end{aligned}$$

and the syndromes defined by

$$\begin{aligned}\boldsymbol{s}_r &= \boldsymbol{y}_r H^T \\ &= (\boldsymbol{y} + \boldsymbol{e}_r)H^T \\ &= \boldsymbol{e}_r H^T.\end{aligned}$$

Note that since both G^{-1} and H^T are polynomial, received errors propagate in the noisy estimates and syndromes for only a finite time. If received errors are sparse, the noisy estimates $\boldsymbol{x}_r$ correspond quite closely to the actual information sequences $\boldsymbol{x}$, each received error generally giving a finite number of errors in $\boldsymbol{x}_r$. (As Sullivan [6] has pointed out, the number of errors in $\boldsymbol{x}_r$ is sometimes minimized over all inverses by using a suitable pseudo-inverse $\tilde{G}^{-1}$. See the example in Appendix II.) As for the syndromes, by virtue of Theorem 7, absence of errors in the received word from time $t - \mu$ to $t - 1$ is sufficient to guarantee that syndromes at time t and later are due only to errors at time t and later. Since the probability of such an event must be nonzero, any decoder using a feedback-free syndrome generator will eventually be able to get started or resynchronized by repeatedly resetting its internal state to that which would have existed had all past syndromes been zero (the 'syndrome reset' technique).

As with block codes, the syndromes are in one-to-one correspondence with the classes of apparent errors $\boldsymbol{e}_a$ (defined as $\boldsymbol{e}_a = \boldsymbol{y}_r - \boldsymbol{y}$) between which the codeword estimator must choose, since $\boldsymbol{e}_a H^T = \boldsymbol{s}_r$ if and only if $\boldsymbol{y} = \boldsymbol{y}_r - \boldsymbol{e}_a$ is a codeword, which is a necessary condition for a codeword estimator output. Another way of saying this is that H^T is a many-to-one homomorphism from n-tuples to $(n - k)$-tuples whose kernel is the code generated by G. (As a dual encoder, H executes a one-to-one map in the reverse direction.) Syndromes are especially useful when the channel is such that the received errors $\boldsymbol{e}_r$ are independent of which codeword was actually sent, because then the decoding rule can be simply a map from syndromes $\boldsymbol{s}_r$ to apparent errors $\boldsymbol{e}_a$.

Finally, we observe that for any code a syndrome former H^T can be realized with the same number of memory elements as a minimal encoder since H^T is just the dual encoder H with inputs and outputs exchanged. (Theorem 8 can be used to prove this.)

Appendix II

Systematic Encoders

We have settled on minimal encoders as the canonical encoders for convolutional codes. In general a minimal encoder is nonsystematic; that is, the information sequences do not in general form part of the codeword. On the other hand, convolutional coding theory and practice have been almost exclusively concerned with systematic encoders. We believe that this historical happenstance is due mostly to misconceptions: false analogies with block codes, apprehensions about error propagation, feelings of insecurity about not having the true information bits buried somewhere in the received data. However, there are some situations in which a systematic code is definitely to be preferred.

Costello [8] was apparently the first to notice that every convolutional encoder is equivalent to a systematic encoder that may itself contain feedback, but which has a delay-free feedback-free inverse. In this Appendix we extend Costello's result to show that there is such an encoder that is realizable with the same number of memory elements as the minimal encoder, and thus also merits the label 'canonical.' The proof requires an appeal to realizability theory, whose main theorem we state, with a sketch of its proof. As a corollary we obtain a generalization of the result of Bussgang that every convolutional encoder is equivalent to a feedback-free systematic encoder over a decoding constraint length. We conclude with a comparison of our two classes of canonical encoders, minimal and systematic.

Realizability Theory: Main Theorem

We will first make a slight digression into realizability theory to pick up its main theorem [11].

Theorem 8: Let G have invariant factors γ_i with respect to $F[D^{-1}]$. Assume $\gamma_i \neq 0$, $1 \leq i \leq k$, and let $\gamma_i = \alpha_i/\beta_i$, where α_i and β_i are polynomials in D^{-1} with no common factors and β_i is monic; let μ_i be the degree of β_i. Then the state space Σ_G has dimension $\mu = \Sigma\mu_i$, and G is realizable with μ memory elements, but with no less than μ.

Sketch of Proof: That G needs at least μ memory elements for realization can be shown by exhibiting μ independent states and thus proving that the state space has dimension of at least μ. Let G have an invariant factor decomposition $G = A\Gamma B$ with respect to $F[D^{-1}]$, and consider the μ anticausal inputs

$$D^{-j}\boldsymbol{\varepsilon}_i A^{-1} \qquad 1 \leq i \leq k, 0 \leq j \leq \mu_i - 1;$$

these lead to the μ states

$$\begin{aligned}\boldsymbol{s}_{ij} &= D^{-j}\boldsymbol{\varepsilon}_i A^{-1} GQ \\ &= D^{-j}\boldsymbol{\varepsilon}_i \Gamma B Q \\ &= D^{-j}\gamma_i \boldsymbol{b}_i Q \qquad 1 \leq i \leq k, 0 \leq j \leq \mu_i - 1.\end{aligned}$$

Any linear combination over F of these states $\Sigma_i\Sigma_j\psi_{ij}\boldsymbol{s}_{ij}$

can be written in terms of the polynomials in D^{-1}

$$\psi_i = \sum_{j=0}^{\mu_i - 1} \psi_{ij} D^{-j}$$

as

$$\sum_i \sum_j \psi_{ij} s_{ij} = \sum_{i=1}^{k} \psi_i \gamma_i b_i Q.$$

We see that such a combination can be equal to 0 only if the $\psi_i \gamma_i b_i$ and hence (since B has an anticausal inverse) the $\psi_i \gamma_i$ are anticausal sequences, i.e., polynomials in D^{-1}, but $\psi_i \alpha_i / \beta_i$ is a polynomial in D^{-1} only if β_i divides ψ_i (since gcd $(\alpha_i, \beta_i) = 1$), which is impossible since the degree of ψ_i is no greater than $\mu_i - 1$ and hence less than the degree of β_i (as polynomials in D^{-1}). A slight extension of the argument would show that all states are linearly dependent on these and hence that the state space has dimension exactly μ.

To exhibit a realization, it is necessary only to construct a linear circuit whose physical states correspond to the abstract states above, and then to arrange that each unit input set the circuit into the appropriate state, as well as give the output at time 0 directly. Such a realization can be constructed from k feedback shift registers of lengths μ_i, with feedback connections given by the denominator terms β_i, as was illustrated in Fig. 2. As the register contents at time 1^- range through all possibilities, a μ-dimensional space, the 'output' of these registers ranges through all linear combinations of sequences of the form α_i / β_i, where α_i is a polynomial in D^{-1} of degree less than or equal to $\mu_i - 1$, and the 'output' is any linear combination of the contents of these registers. Now it is not hard to see that an encoder G with invariant factors α_i / β_i with respect to $F[D^{-1}]$ can be realized by a circuit of the type illustrated in Fig. 6, where C_1 is a purely combinational circuit that generates μ_i vectors to be added (linearly over F) to the register contents, and C_2 is a purely combinational circuit that forms a linear (over F) combination of the register contents and the current inputs to give the current outputs. The only delay elements are therefore in the feedback shift registers, and their number is $\Sigma \mu_i = \mu$.

Q.E.D.

Canonical Systematic Encoders

Block codes are generated by nonsingular $k \times n$ matrices with elements in some field. Any such matrix has some $k \times k$ submatrix that is nonsingular and therefore has an inverse; premultiplication by this inverse yields an equivalent (same row space) generator matrix in systematic form, that is, with the identity matrix as a submatrix. Thus every block encoder is equivalent to a systematic encoder; consequently systematic encoders have universally been taken as the canonical class of block encoders ([9], chs. 2 and 3).

As we have seen, the elements of the generator matrix

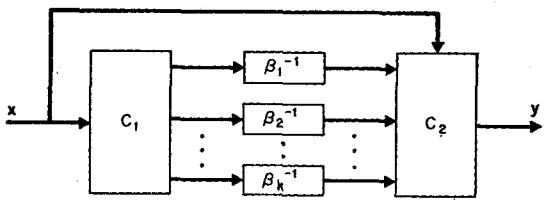


Fig. 6. Canonical realization of (n, k) encoder.

G of a convolutional code are in the ring $F_{rz}(D)$ of realizable functions. A realizable $k \times k$ matrix has a realizable inverse if and only if its determinant is a nonzero realizable function with delay zero (from our general results on inverses, or Cramer's Rule). Now every convolutional encoder is equivalent to a basic encoder, and any basic encoder must have some $k \times k$ submatrix with a determinant that is a nonzero polynomial not divisible by D, else all $k \times k$ subdeterminants would be divisible by D and the encoder would not be basic. Premultiplication by the realizable inverse of such a submatrix yields an equivalent realizable encoder in systematic form. Hence Costello's result, which follows.

Theorem 9: Every convolutional encoder is equivalent to a systematic encoder.

There is some freedom in the choice of the k columns of the equivalent systematic encoder that are to be the k unit column vectors ϵ_i; once these columns are specified, assuming the ϵ_i are in their natural order, then the systematic encoder generating a given code is unique. Such an encoder G has a trivial delay-free feedback-free inverse, namely, the matrix G^{-1} with unit row vectors in the rows corresponding to the unit columns of G, and zeroes elsewhere, which simply strips off the k transmitted sequences that are identical to the information sequences. As in the block-code case, a dual systematic $n \times (n - k)$ encoder H^T can be formed by putting unit row vectors in the rows not corresponding to unit columns of G, and filling in the remaining $k \times (n - k)$ matrix with the negative of the remaining $k \times (n - k)$ matrix of G. (see [9], section 3.2.) (From this fact alternate proofs of the results of Appendix I can be constructed.)

To be canonical, an encoder must be realizable with μ memory elements, where μ is the minimum number of memory elements in any equivalent encoder, hence the number in an equivalent minimal encoder. Theorem 10 shows that systematic encoders have this property as well.

Theorem 10: Every systematic encoder G is canonical; that is, can be realized with the same number μ of memory elements as an equivalent minimal encoder.

Proof: From Lemma 5, dim $\Sigma_G = \mu$, since G has a polynomial inverse. From Theorem 8, G can therefore be realized with μ memory elements.

Q.E.D.

We should note that the canonical realization of even a feedback-free systematic encoder may not be the obvious

realization; for example, it is well known that the most efficient realization of a conventional systematic rate-$(n-1)/n$ code, $n > 2$, with maximum generator degree ν, is Massey's [14] type-II encoder in which a single length-ν register forms all parity bits, as in Fig. 7.

Any practical decoder has some decoding constraint length λ, namely, the number of time units of delay between the time t any given information symbol is sent and the time $t + \lambda$ that a decision on that symbol must be generated by the decoder. If all previous symbols are known and the channel is memoryless, then the decoding decision may be based solely on the received word in the interval from t to $t + \lambda$, with the effects of previous symbols subtracted out (by linearity); by constancy, the decoding problem can thus be reduced to finding the initial symbols in a sequence that "starts" at time 0 on the basis of received data in times 0 through λ. Clearly only the structure of the codewords out to time λ, or modulo $D^{\lambda+1}$, enter into such a decision. We say that two encoders with the same codewords modulo $D^{\lambda+1}$ are equivalent over a decoding constraint length. Bussgang [15] observed that any $(n, 1)$ encoder is equivalent to a systematic feedback-free encoder over a decoding constraint length. Extension of this result is an easy corollary to Theorem 9.

Corollary 3: Every encoder is equivalent to a systematic feedback-free encoder over a decoding constraint length.

Proof: By Theorem 9 every encoder is equivalent to a systematic encoder, a fortiori equivalent modulo $D^{\lambda+1}$. Any realizable sequence is congruent modulo $D^{\lambda+1}$ to a polynomial of degree λ, namely, the first $\lambda + 1$ terms in such a sequence. Hence such a systematic (in fact any) encoder is equivalent to a polynomial encoder modulo $D^{\lambda+1}$. Q.E.D.

(Note that when the decoder estimates inputs at time t on the basis of received data in times t through $t + \lambda$, we can restrict our attention to the set S of codewords modulo $D^{\lambda+1}$ generated by inputs that start at time 0 or later. It may happen that two encoders equivalent over λ by our definition may not have the same sets S, due to delay in the encoder; for example $[1, 1 + D]$ appears in S when $G = [1, 1 + D]$ but not when $G = [D, D + D^2]$. This point is discussed by Costello [8] under the rubric "causal dominance;" it does not arise whenever G has a zero-delay inverse.)

Two further caveats must be entered: 1) if λ is large, the equivalent systematic encoder may be much more complex than the original encoder; 2) two encoders equivalent over a decoding constraint length behave identically only until a decoding error is made, when their error propagation characteristics may be very different.

Comparison of Minimal and Systematic Encoders

We have seen that either minimal or systematic encoders may be taken as a canonical class of convolutional encoders. Here we discuss the relative merits of each from both theoretical and practical viewpoints.

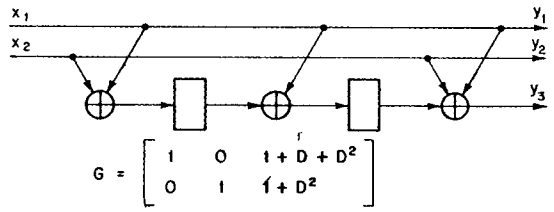


Fig. 7. Canonical realization of rate-2/3 systematic encoder.

Some codes can be generated by encoders that are both minimal and systematic, or at least feedback-free and systematic. Bucher and Heller [16] have shown that such codes are inferior to general codes of the same constraint length, in the sense that the error probability on memoryless channels with maximum-likelihood decoding is greater over a random ensemble of such codes than over the general ensemble. Bucher [17] has also shown that still further degradation in performance occurs when sequential decoding is used to decode such codes, whereas it is known that sequential decoding of general codes is asymptotically as good as maximum-likelihood decoding. Hence one does not want to confine one's attention to feedback-free systematic codes.

Both systematic and minimal encoders have delay-free feedback-free inverses. However, while the existence of former inverse is obvious, that of the latter can be demonstrated only with the aid of some algebra, and has not in the past been generally appreciated. One suspects that the main reason that nonsystematic encoders have heretofore not been used is ignorant fear of error propagation. Such fears are largely groundless, for a feedback-free inverse guarantees no catastrophic error propagation, while we have seen in Appendix I that one can find feedback-free syndrome formers as well, which in most situations can be used to ensure against ordinary error propagation.

Systematic encoders seem to be reassuring to some people by virtue of preserving the original information sequences in the codewords. The thought appears to be that at least if the decoder doesn't work, the information bits will still be there. One situation in which this consideration has real force is a broadcast situation in which information and parity sequences are sent over separate channels, and only some of the receivers have decoders, while the rest depend on the information sequence alone. If, on the other hand, all transmitted sequences are combined into one serial stream, then in order to pick out the information bits, the receiver at least has to establish phasing, and if it can do this, it is hard to see why it can not also make the feedback-free inverse linear transformation G^{-1} to recover noisy estimates of the signal. It is true that when the channel error probability is p, the error probability in these noisy estimates will be at its minimum of p when the encoder is systematic. However, when the decoder *is* working, we have seen that only minimal encoders uniformly associate short output error sequences

with short error events; in general short error events may result in many output errors with systematic encoders.

Example: Minimal encoder = $[1 + D + D^2, 1 + D^2]$. Pseudo-inverse with delay D: $[1, 1]^T$. Output error probability when decoder is not working: $2p$(for p low). Most likely error event (only codeword of weight 5): $[1 + D + D^2, 1 + D^2]$. Output errors per most likely error event: 1.

Equivalent systematic encoder = $[1, 1 + D^2/1 + D + D^2]$. Inverse: $[1, 0]^T$. Output error probability when decoder is not working: p. Most likely error event: same as above. Output errors per most likely error event: 3.

It appears that if we expect the decoder to be working, we should select a minimal encoder, while if we expect it not to be, we should select a systematic encoder. This is not as frivolous as it sounds; in a sequential decoder, for example, actual (undetected) error events can be made extremely rare, the decoder failures instead occurring at times when the decoder has to give up decoding a certain segment because of computational exhaustion. During these times the decoder must put out something, and the best it can do is generally to put out the noisy estimates obtained directly from the received data, the errors in which will be minimized if the encoder is systematic. A systematic encoder (with feedback) might therefore be a good choice for a sequential decoder, depending on the resynchronization method and the performance criterion. On the other hand, a maximum-likelihood decoder (Viterbi algorithm) is subject only to ordinary error events and as a consequence should be used with a minimal encoder.

As a final practical consideration, the feedback in the general systematic encoder can lead to catastrophes if there is any chance of noise causing a transient error in the encoding circuit.

From a theoretical point of view, minimal encoders are particularly helpful in analyzing the set of finite codewords, as we saw in the main text. The fact that they are a basis for the $F[D]$-module of all such codewords means that we can operate entirely in $F[D]$, which is convenient, although throughout this paper we have seen the utility of considering larger rings. The outstanding theoretical virtue of systematic encoders is that under some convention as to which columns shall contain the identity matrix, there is a unique systematic encoder generating any code. Thus systematic encoders are most suited to the classification and enumeration of codes. Our taste is indicated by the relative placement of minimal and systematic codes in this paper, but clearly there are virtues in each class.

V. Acknowledgment

The work of Prof. J. L. Massey and his colleagues at Notre Dame, particularly the result of Olson [7], was the initial stimulus for the investigation reported here, and should be considered the pioneering work in this field. The principal results were at first obtained by tedious constructive arguments; subsequently Prof. R. E. Kalman was good enough to send along some of his work, which pointed out the usefulness of the invariant factor theorem in the guise of the Smith–McMillan canonical form, and which consequently was of great value in simplifying and clarifying the development. The close attention of Dr. A. Kohlenberg and Prof. J. Massey to the final draft was also helpful.

References

[1] J. K. Omura, "On the Viterbi decoding algorithm," *IEEE Trans. Information Theory*, vol. IT-15, pp. 177–179, January 1969.
[2] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Information Theory*, vol. IT-13, pp. 260–269, April 1967.
[3] J. L. Massey and M. K. Sain, "Codes, automata, and continuous systems: Explicit interconnections," *IEEE Trans. Automatic Control*, vol. AC-12, pp. 644–650, December 1967.
[4] ——, "Inverses of linear sequential circuits," *IEEE Trans. Computers*, vol. C-17, pp. 330–337, April 1968.
[5] M. K. Sain and J. L. Massey, "Invertibility of linear time-invariant dynamical systems," *IEEE Trans. Automatic Control*, vol. AC-14, pp. 141–149, April 1969.
[6] D. D. Sullivan, "Control of error propagation in convolutional codes," University of Notre Dame, Notre Dame, Ind., Tech. Rept. EE-667, November 1966.
[7] R. R. Olson, "Note on feedforward inverses for linear sequential circuits," Dept. of Elec. Engrg., University of Notre Dame, Notre Dame, Ind., Tech. Rept. EE-684, April 1, 1968; also *IEEE Trans. Computers* (to be published).
[8] D. J. Costello, "Construction of convolutional codes for sequential decoding," Dept. of Elec. Engrg., University of Notre Dame, Notre Dame, Ind., Tech. Rept. EE-692, August 1969.
[9] W. W. Peterson, *Error-Correcting Codes*. Cambridge, Mass.: M.I.T. Press, 1961.
[10] C. W. Curtis and I. Reiner, *Representation Theory of Finite Groups and Associative Algebras*. New York: Interscience, 1962, pp. 94–96.
[11] R. E. Kalman, P. L. Falb, and M. A. Arbib, *Topics in Mathematical System Theory*. New York: McGraw-Hill, 1969, ch. 10.
[12] R. E. Kalman, "Irreducible representations and the degree of a rational matrix," *J. SIAM Control*, vol. 13, pp. 520–544, 1965.
[13] B. McMillan, "Introduction to formal realizability theory," *Bell Sys. Tech. J.*, vol. 31, pp. 217–279, 541–600, 1952.
[14] J. L. Massey, *Threshold Decoding*. Cambridge, Mass.: M.I.T. Press, 1963, pp. 23–24.
[15] J. J. Bussgang, "Some properties of binary convolutional code generators," *IEEE Trans. Information Theory*, vol. IT-11, pp. 90–100, January 1965.
[16] E. A. Bucher and J. A. Heller, "Error probability bounds for systematic convolutional codes," *IEEE Trans. Information Theory*, vol. IT-16, pp. 219–224, March 1970.
[17] E. A. Bucher, "Error mechanisms for convolutional codes," Ph.D. dissertation, Dept. of Elec. Engrg., Massachusetts Institute of Technology, Cambridge, September 1968.